RADC-TR-81-207
Final Technical Report
July 1981

INTELLIGENCE PROCESSOR PERFORMANCE ANALYSIS (IPPA)

Measurement Concept Corporation

Jim Cantrell
Roger Poland
Jim Labout - Punker Ramo Corporation

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-81-207 has been reviewed and is approved for publication.

APPROVED: John E. Frank

JOHN E. FRANK
Project Engineer

APPROVED: John N. Entzminger Jr.

JOHN N. ENTZMINGER
Technical Director
Intelligence and Reconnaissance Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (IRDE) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-81-207	2. GOVT ACCESSION NO. AD-A108 794	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) INTELLIGENCE PROCESSOR PERFORMANCE ANALYSIS (IPPA)		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report 29 Feb 80 - 31 Mar 81
		6. PERFORMING ORG. REPORT NUMBER N/A
7. AUTHOR(s) Jim Cantrell Roger Poland Jim Labout - Bunker Ramo Corporation		8. CONTRACT OR GRANT NUMBER(s) F30602-80-C-0099
9. PERFORMING ORGANIZATION NAME AND ADDRESS Measurement Concept Corporation 1721 Black River Blvd. Rome NY 13440		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62702F 45941643
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (IRDE) Griffiss AFB NY 13441		12. REPORT DATE July 1981
		13. NUMBER OF PAGES 244
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: John E. Frank (IRDE)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Monitor (s)
Hardware Monitors
AN/GYQ-21(V)
Computer Performance Evaluation
PDP-11's

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report has documented the functional design that was developed for an AN/GYQ-21(V) hardware monitor.

The report provides a concept of monitoring to place the development of the hybrid monitor in its proper perspective.

The report defines the measurement set the monitor will measure. The functional design includes both the software and hardware components with the rationale for their use to be able to monitor the measurement

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

313

set selected.

The software specifications include tasks such as coordination between the host processor and monitor, measurements, and measurement reporting. Foremost in the designer's mind has been the ease of user interface with the monitor.

The hardware specifications include such areas as packaging, power requirements, EMI/RFI emissions, Qualifier Logic, and special modules.

The report includes a theory of operations that references the design and specifications and provides an insight into functional breakout by measurement requirements.

All of the technical and software/organizational techniques presented in the report are within the state-of-the-art. As a result, the microprocessor based performance monitor possessing an extraordinary degree of applicability to all potential users is eminently achieveable.

In short, it is believed that the simplicity of staged measurement handling, maximum utilization of nonsensitive probe interfaces such as the 21(V) busses, and the integration of passive hardware, active hardware, software, and cooperative hybrid measurement/control techniques provide RADC and the end user with a highly useful, low risk product at a cost commensurate with the complexity of the user requirement.

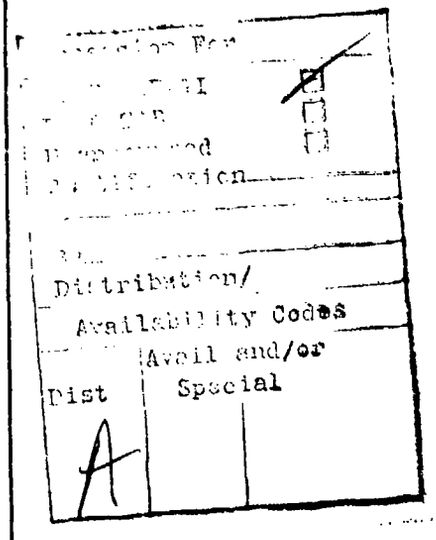


TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION

Measurement Concept Corporation (Mc2) submits this Final Technical Report covering the results of the Intelligence Processor Performance Analysis (IPPA) study effort. This project involved engineering services to validate the operation and characteristics of a hardware/software (hybird) performance monitor for the AN/GYQ-21(v), and to produce detailed engineering specifications for a very low cost, expandable Prototype Model. During the effort Mc2 and its subcontractor, Bunker Ramo Corporation, have produced designs of the software and hardware for the IPPA. Those designs are presented within this document. The balance of this section presents the overview of the hybrid monitor.

The design is sectioned into two volumes. The first volume provides all software and hardware design elements and the second volume contains manufacturer reference material for devices used

Chapter 2 describes the configuration of the software modules required for IPPA. This includes software to intercept data within the host and to relay that information to the IPPA hardware, plus the software necessary to process the data in the hardware's microprocessor.

Chapter 3 describes the design of the hardware portion of IPPA, and explains the basic hardware units that constitute the monitor. This is expanded in Chapter 4 - Theory of Operation - which also contains schematics of the design. A parts list and timing diagrams appropriate to the design appear in Chapter 5. A list of selected data specifications appropriate to the design appear in Chapter 6.

1.1 Overview

The overall hardware/software (hybrid) monitoring approach involves hardware probes which are supplemented by software "probes" in the monitored "21(v)" CPU. In this approach data from both probe types are processed together to detect events that could not be detected by independent hardware and software monitors.

The software events, such as task initialization and issue of executive directives are detected by a host resident, interceptive, system-transparent software monitor based upon proven techniques that have been demonstrated by Mc2 in its software monitor. To minimize host artifact, very little processing beyond event identification is to be performed by host resident software. Instead, the coded "event number" is to be passed to the microprocessor in the hardware monitor through a register set addressed in the host's "external" address page. Thus the software probe's interface to the microprocessor looks like a peripheral controller port.

There are two software probe data transfer paths to the hardware monitor; one will interrupt the hardware monitor's microprocessor's event processing logic, the other will directly modify an event "mask" register in the hardware monitor. This mask is used by relatively high speed event detection logic to determine, among other things, the occurrence of software state-related events.

Because of the 21(v) bus structures, most (but not all) of the information needed by system engineers and developers is available without need for discrete probes. This design approach makes maximal use of bus-derived information and thereby reduces both cost and interference with normal host utilization. Whenever possible, bus data common to all 21(v) mainframes (i.e., Unibus and Console) is used. Mass bus data (11/70 only) is collected by a separate board which is not present in lower level 21(v) mainframes.

The hardware probe concept involves four levels of detection/processing as depicted in Figure 1-1. At Level 1, very high speed signals are preprocessed to determine "logical" level, slower speed signals that are used for event detection and counting at Level 2. The IPPA microprocessor processes accumulations of Level 2 data and a few critical Level 2 events at the more leisurely pace (100 microseconds) of Level 3 and completes more complex data organization functions at Level 4 (milliseconds or more).

1.2 Measurement Methodology

This section provides a brief overview of the IPPA mechanisms invoked to satisfy the measurement set provided in Annex I to the SOW. This section does not attempt to identify specific hardware/software IPPA components to be used for each measurement but instead deals with each measurement on a conceptual basis. Transfer of concept to design takes place in later sections of this document.

1.2.1 Group A Measurements - CPU Performance

1.2.1.1 Task Time - CPU

A software probe intercepts the host upon completion of an active task list scan. If the task to be given control is a task being measured, the software probe so informs the hardware monitor. The event causes a time stamped microprocessor interrupt and/or it directly enables a gate. Task exits occur via EMT's or interrupts. EMT's are detected by the host software probe and the information is passed to the microprocessor in a manner similar to that above. Interrupts are directly detected via INTR by the IPPA hardware probes and are used to interrupt the microprocessor at high priority so that the task timing

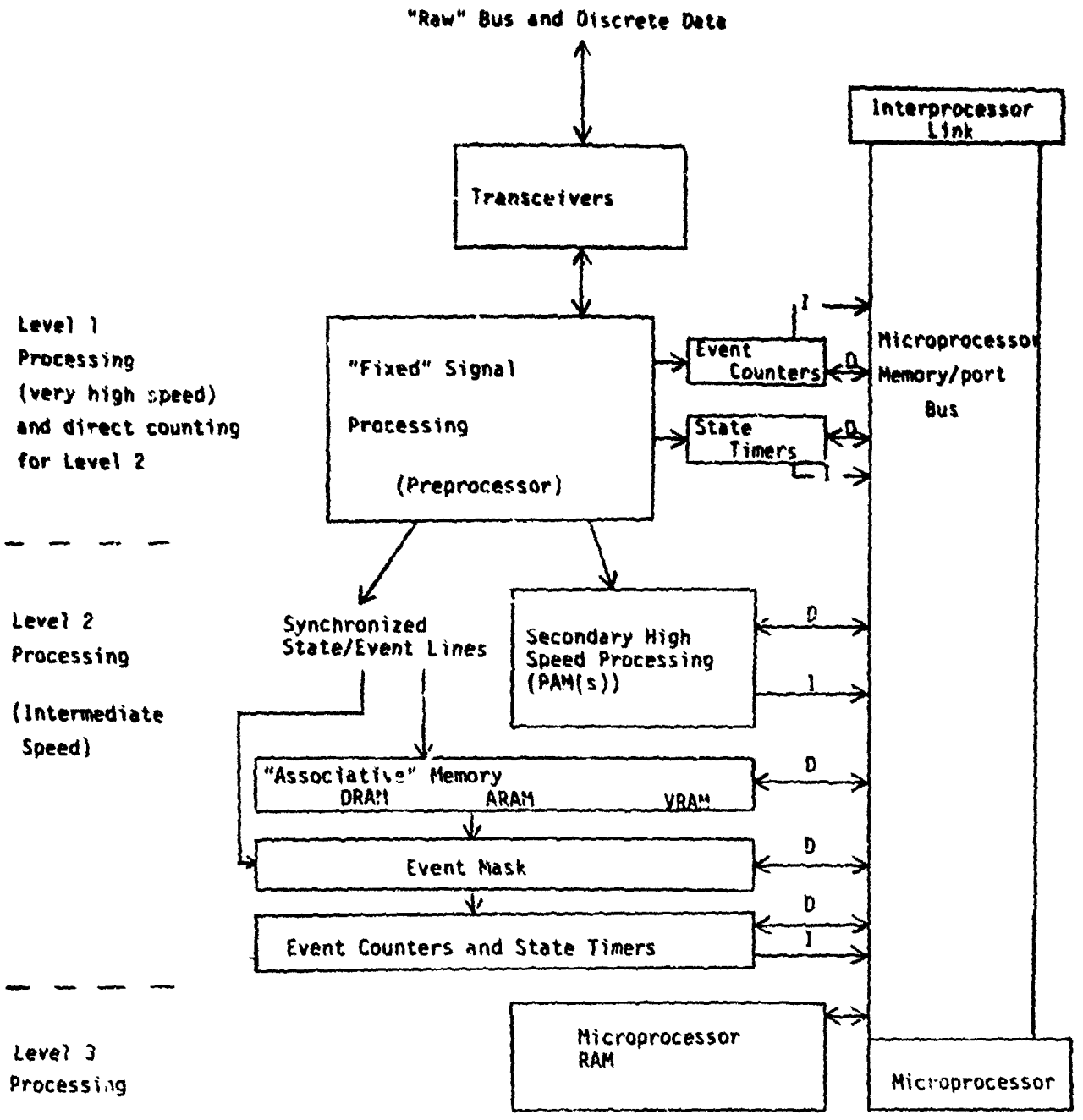


Figure 1-1 IPPA Hardware Probe Concept

can be captured. Returns from EMT processing and interrupt service routines are intercepted by the software probe which passes the information back to the microprocessor.

1.2.1.2 Event Time - CPU

The set of possible events is very large. Software events include task start and task stop. Hardware events can be determined by combinations of CPU priority and mode, bus cycle type, BR/NPR status, instruction fetch virtual address, and value of data transferred.

Events able to "bump" a counter or change a clock status will also be able to interrupt the microprocessor. However, some of the more esoteric combinations of signals (BR/NPR status combined with one or more of the other above) would be difficult to detect as it is unlikely that all signals will be brought together into one combined selection matrix. Such a combination was therefore not considered. Combinations of virtual address, data value, CPU priority/mode, and bus cycle type will, however, be detected by the design. Event detection will cause a time stamped interrupt of the microprocessor for later (although "near real time") processing in a relatively leisurely fashion.

1.2.1.3 Percent Time Kernel, Supervisor, User Mode

The design provides a set of hardware timers dedicated to this measurement (i.e., the measurement is one of the minimal baseline set). The signals are obtained from the front panel of the CPU. Timer overflow will cause a microprocessor interrupt.

1.2.1.4 Percent Time at Priority Level

The design provides a set of hardware timers dedicated to this

measurement. Priority level information is obtained from discrete probes.

1.2.1.5 Percent CPU Not Busy

This information is obtained from the CPU front panel and directly enables a fast counter dedicated for this purpose.

1.2.1.6 Percent CPU-I/O

The CPU "RUN" state signal obtained from the console panel is "AND"ed with the Unibus cycle or MBC signals to determine when CPU-I/O overlap occurs. This signal enables a timer to obtain the desired measurement.

1.2.2 Group B- Unibus Performance

1.2.2.1 Unibus Acquisition Time

This is construed to mean the time between when a "master" desires the bus and when it actually acquires it. For non-CPU devices, these times are identical to either the SOW Group B items 4 or 5 - Bus request and NPR latency.

1.2.2.2 Unibus Occupancy

This measurement is equivalent to timing the Unibus signal "BBSY". A dedicated hardware timer will be reserved for this measurement.

1.2.2.3 Interrupt Response Latency

This measurement determines the time between a non-CPU master assertion of INTR and the completion of the CPU interrupt sequence.

It is believed that, as the value is unlikely to change for a given CPU unless hardware problems arise, this measurement is best obtained by a maintenance technician with a scope. It is unlikely that this measurement will be performed by the currently conceived IPPA.

1.2.2.4 NPR Latency

NPR latency is the time between a given device's assertion of NPR and the receipt by the device of an NPG. It is believed that the importance of this measurement is related to NPR contention, i.e., the times when NPR remains high for longer periods of time. One case arises when NPR remains high after arbitrator assertion of an NPG. As the NPR/NPG cycle will normally occur rapidly when CPU is master, NPR contention is a strong indicator that one DMA device will need to wait until a second device has completed its transfer. Secondly, assertion of NPR when CPU is not master also can result in a longer latency for the requesting service.

The actual timing of most NPR/NPG pairs is purely a function of the specific hardware and except in cases of contention as above, the time will not vary except if there are machine problems. The large number of "normal" pairs can hide abnormalities unless the abnormalities can be specifically and separately measured.

It is therefore determined that the useful measure is NPR contention and not merely latency.

Because of the high speeds involved, this measurement is to be handled by dedicated IPPA hardware. The initial goal is to measure the total contention time and the number of times contention is detected for all Unibus peripherals. To accomplish this, the IPPA design provides two measurements, either of which may be selected at a given time. In the first mode, total NPR time and the count of NPG's are measured. This

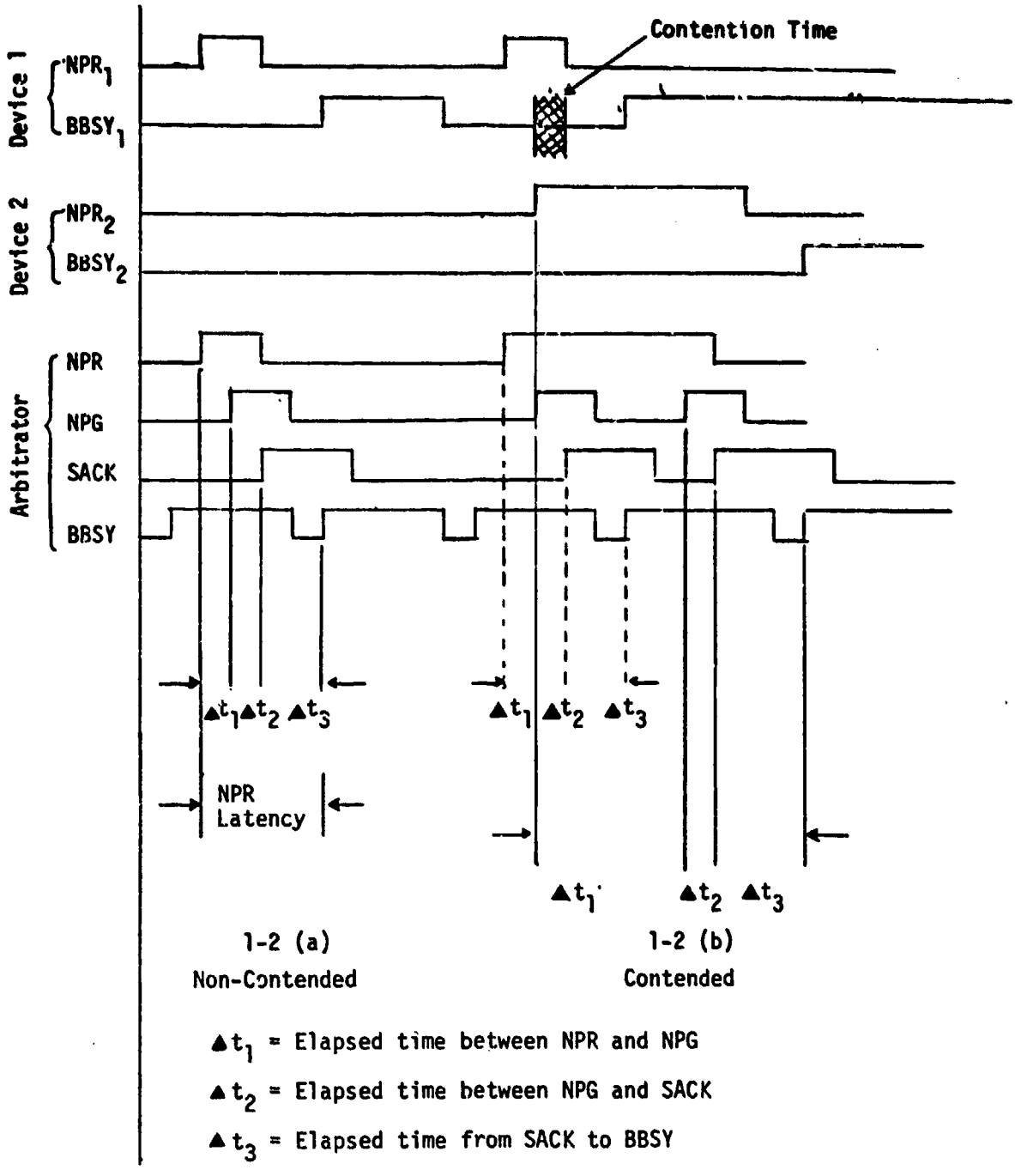
provides an average NPR/NPG latency for use in more complex measures. As illustrated in Figure 1-2, the NPR contention time is determined as the sum of the statistically determined non-contended NPR/NPG latency and, in those cases where NPR remains high after issue of NPG, the time taken by the non-processor device to complete its transfer. Therefore, the count of NPR deassertions provided by the second mode, when subtracted from the number of NPG's (provided in both modes) is a measure of the actual latency caused by contention as it establishes the number of whole non-processor transfer cycles that occur when NPR is contended. The system does not need to be accurate to more than several hundred nanoseconds as we are looking for contention times in excess of one microsecond.

1.2.2.5 Bus Request Latency

Bus Request Latency is nominally defined as the time between device assertion of a bus request and receipt of a bus grant. Unlike NPR's, the BR may not receive a grant if the CPU is running at a priority level equal to or greater than the level of the BR. As a result this measurement is an indicator of device delays caused by execution of code (e.g., an interrupt service routine) having a higher priority level than the bus request.

Whereas NPR latency is a measure of Unibus contention, bus request latency is more a measure of contention for high priority CPU resources. As a result, the desirable timing quantization is more on the order of an instruction cycle than a fraction of a bus transfer cycle. Errors of a few microseconds in an interrupt service routine (ISR) time of 100 microseconds are therefore acceptable.

The IPPA will count the total amount of time that each BR is held high and the number of times it is brought low. As the total number of BG Counts is also known by BR level, the difference in counts is a



▲t_2 is constant in both cases.

▲t_1 is lengthened in contended cases for the duration a second NPG cannot be issued. This time is approximately that between the first bus grant and the dropping of BBSY by the master at time of contention. It is therefore, an average bus acquisition time.

▲t_3 is also lengthened by an average bus acquisition time.

Figure 1-2 NPR Arbitration

measure of the number of times for each BR level that BR latency is raised by multiple requests at the same BR level. The accumulated BR time divided by the count of BG's is the average latency. By using the average latency at very low request levels as a baseline for noncontention BR service times, we may remove this baseline from the total measured times. This leaves only contention time which may be divided by the number of contentions to yield average contention time. It is noted that total contention time is, by itself, a valuable measure.

As the high performance peripherals will be at high BR levels and as they are of most interest, this scheme is expected to provide the necessary information.

1.2.2.6 UMR Utilization

UMR's are infrequently modified and then only through host software. Their state will be made available also by host software means.

1.2.2.7 Number of Transfers per Second

A separate counter is dedicated to counting the MSYN/SSYN pairs. The counter may be read by the microprocessor.

1.2.2.8 Read/Write Counts

The bus signal lines C0 and C1 are monitored. Either slave writes or slave reads are counted so that, with total transfers, both are known.

1.2.3 Group C - Peripheral Performance

1.2.3.1 Memory References by Range

This is a mixed item that includes both CPU and peripheral performance measures. For non-task-oriented references, physical addresses are obtained from the console connection. The associative address-processing logic of the IPPA Address Associate Ram (ARAM) performs the range partitioning and is supported by "Read/Write", "Fetch", and similar signals that "mask" the partition events.

1.2.3.2 Memory Range by Task

A key ingredient of the IPPA design is the use of virtual addresses for all task-oriented memory range measurements:

- o Programmers do not know physical addresses when assembling/ task building code.
- o Programs are not always loaded at the same physical addresses.

Using virtual addresses provided by the CPU console and task activation/ deactivation data provided by the software probes, the same "ARAM" logic used for 1.2.3.1 is applicable to this measurement.

1.2.3.3 R/W to Peripheral Controllers

All R/W commands and programmed I/O appear on the Unibus. Addresses are unique for each controller/port and can be matched by several methods including the ARAM and "smart" controller of the Peripheral Activity Module (PAM). Data is also available so that the "go" bit of a standard CSR register can be detected. Programmed I/O is a special case of a memory partition one word wide so that all such I/O can be counted via ARAM output.

1.2.3.4 NPR's/Controller and Unit

This is a most difficult measurement to make without discrete probes to each peripheral. The IPPA design concept minimizes such probes for emission reduction and operational reasons so that this area requires more sophistication than most others.

In short, a high speed, bit slice micro is tasked with intercepting word count and buffer addresses for selected peripherals. When ready to transfer data, a Unibus peripheral controller becomes bus master via the NPR/NPG and puts a buffer address on the Unibus so that the monitor can compare it to the buffer address range of the selected peripheral that was previously intercepted.

As each Massbus peripheral may be uniquely identified by its location on the bus, no such sophistry is needed for these cases.

1.2.3.5 INTR's by Controller/Unit

This is accomplished by the VECR circuitry which detects the INTR and determines the vector address placed on the Unibus by the controller. As unit selection is also known, the INTR may be assigned to the selected unit.

1.2.3.6 Memory/Peripheral Transfers/Seconds

The above paragraphs show how programmed I/O, CPU operations, NPR transfers, and Massbus transfers can individually be detected. This measurement is obtained by counting the total of such transfers and dividing the count change by the time interval over which the count changed.

1.2.3.7 Transfer Service Time

As the "GO" event and INTR are both detected by the IPPA hardware, a

straightforward time stamp of these events enables the microprocessor to do the necessary calculation.

1.2.3.8 Number of On-Line Terminals

This will be handled by host software elements.

1.2.3.9 Number of Units Busy

This measurement loses value unless it is an instantaneous "snapshot" of system activity. It is therefore suggested that in lieu of this measurement, we determine the percentage of time that a unit is busy.

For DMA devices this is the same as paragraph 1.2.3.7 where a measuring count is maintained. For non-DMA devices, (e.g., a printer) no such measure is available without discrete probes. We therefore suggest that transfer rate be used as a measure of unit "business" (see paragraph 1.2.3.3) in this case.

1.2.3.10 Disk Head Motion/Position

This information will be detected by the peripheral activity module (PAM), which monitors the I/O page and associated registers of DMA devices such as track/sector addressing.

1.2.3.11 Disk Start/Stop Times

The IPPA design will not provide this measurement. It is believed this data is better obtaired through computer operator procedure logs.

1.2.3.12 Cache Acquisition Latency/Contention

Multiple requests to the memory logic are detected via "AND" gates.

When multiple requests are active, the resultant signal will enable a counter and/or timer. Total contention time divided by the number of contended accesses provides average contention time.

1.2.3.13 Cache Hit Rates

The cache slow cycle signal is monitored and the number of such cycles is counted. As the total number of cache read/writes is also monitored, the hit rate is determined by simple ratio.

1.2.3.14 Cache Memory Transfer Rate

See paragraph 1.2.3.6.

1.2.3.15 MBC Percent Busy

See paragraph 1.2.3.9.

1.2.3.16 MBC Transfer Rate

See paragraph 1.2.3.4

1.2.4 Program Performance

1.2.4.1 Task Request Rate

This is obtained by monitoring the request directive in the executive. It is accomplished with the software probe which passes the coded data to the microprocessor for storage and accumulation.

1.2.4.2 I/O Request by Task/Device

The software probe intercepts "QIO" directives.

1.2.4.3 Number of Active Tasks in Memory

This is accomplished by software in the host that is activated at regular intervals to sample the status of the host OS.

1.2.4.4 Memory Allocation

This data is acquired by intercepting the system's memory management routines, using the routines' inputs, and passing them to the microprocessor.

1.2.4.5 Node Status

The information on node allocation and deallocation is performed in a similar manner to paragraph 1.2.4.4

1.2.4.6 Instructions/Second

This is obtained by counting the number of "Instruction Fetches" in a given period of time. The signal is obtained from a discrete probe. The counter is armed/disarmed by software events such as task start/stop.

1.2.4.7 Instruction Distribution

This option will be provided by a separate program that will run the program one instruction at a time and collect the instruction distribution. This software and hybrid monitoring should not be done at the same time.

1.2.4.8 Floating Point Instructions

The IPPA design provides for limited matching of data via the Data

Associative RAM (DRAM). As instructions appear on the data lines during the instruction fetch, the limited number of floating point instructions can be detected and counted.

1.2.5 Fault Detection/Isolation

1.2.5.1 Slave Faults

A slave fault (no Slave Synch acknowledgement) causes CPU trap to address 4. Software can intercept the normal flow to obtain desired information.

1.2.5.2 Trace

Software will be provided to obtain this measurement offline from a monitoring run.

1.2.5.3 Error Counts

These errors are best obtained by the error log programs provided by Digital Equipment Corporation. However, the programmable nature of the PAM makes sampling of a device's error bit at INTR time relatively straightforward.

1.2.5.4 Unclaimed NPG, BG

It is our understanding that the system fails safe under conditions where NPR/BR does not drop with NPG/BG except that the instruction execution rate would significantly drop and/or, if the failed device is close to the arbitrator, devices further out on the Unibus could not be serviced. In case one, the lowered rate is detected. In case two an error is detected (paragraph 1.2.5.3) or the system "hangs" in a state known to the peripheral activity module.

1.2.5.5 Memory Out of Bounds

See paragraph 1.2.5.1.

1.2.5.6 Odd Addrress Error

See paragraph 1.2.5.1

1.2.5.7 Multiple NPR's

See paragraphs 1.2.5.4 and 1.2.2.4.

1.2.5.8 Multiple BR's

See paragraphs 1.2.5.4 and 1.2.2.5.

1.2.5.9 EMT Service Time

The software probe intercepts issued EMT's and can count specific EMT's by task, all EMT's by task, total occurrances of specific EMT's or total occurrances of all EMT's. IOT's are handled similarly.

1.2.5.10 Event Driver Trap

This measurement is obtained in the same manner as paragraph 1.2.5.1.

1.2.5.11 Power "Glitches"

Because of their multiplicity and spatial dispersion, there is no intent at this time to monitor the actual power lines.

1.2.5.12 Power Failure

Upon a significant enough "glitch" in power, the bus ACLO and DCLO lines are changed and the CPU enters a power fail trap. Detection via the trap is easy. Specific processing to be accomplished includes notification to the microprocessor but processing must be kept to a minimum to allow normal OS-provided graceful shutdown.

1.3 Data Flow

The flow of data through a monitor session is discussed in this section in three logical phases:

- o Pre-measurement Phase
- o Measurement Phase
- o Post-Measurement Phase

Figure 1-3 depicts the data flow through the phases, and the inter-phase relationships. Figure 1-4 depicts the tree structure of software modules which pertain to each phase.

1.3.1 Pre-Measurement Phase

The pre-measurement phase involves interaction with the user while the user defines the monitor session through a series of menus. The responses from the menus then become inputs to the compiler. The output of the compiler is then input to the measurement session. Figure 1-5 depicts the data from through the pre-measurement phase.

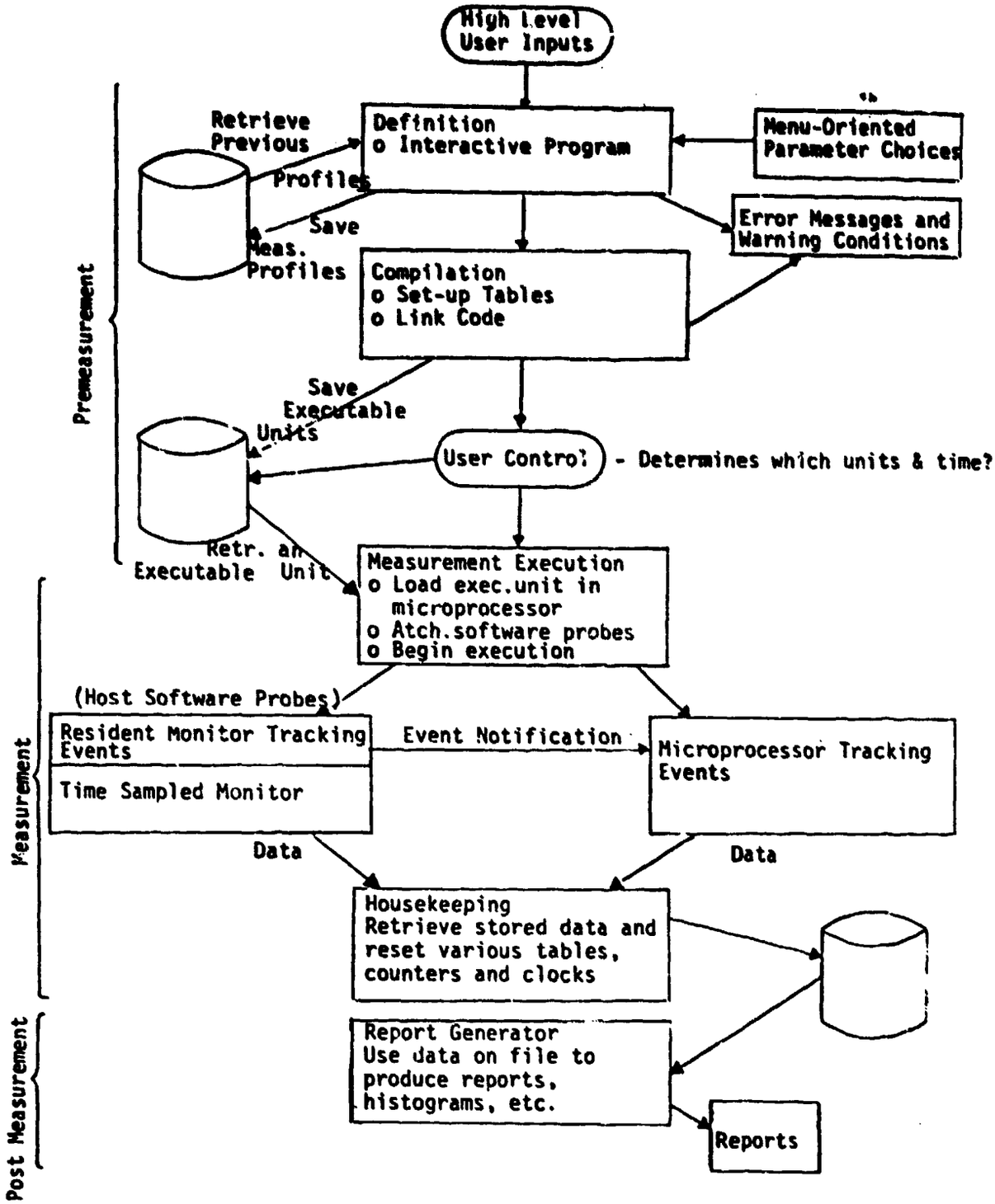


Figure 1-3 Data Flow Chart

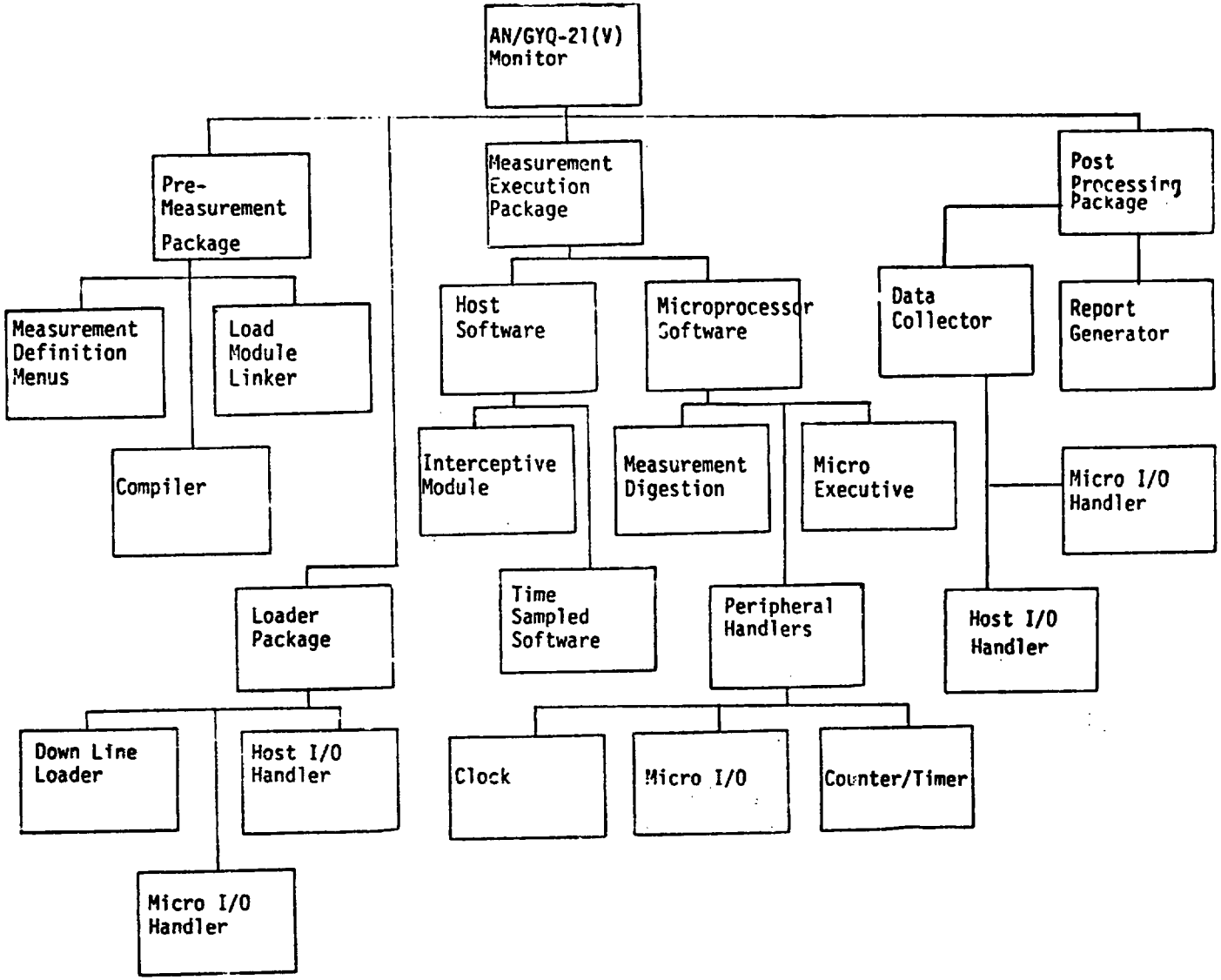


Figure 1-4 Software Modules

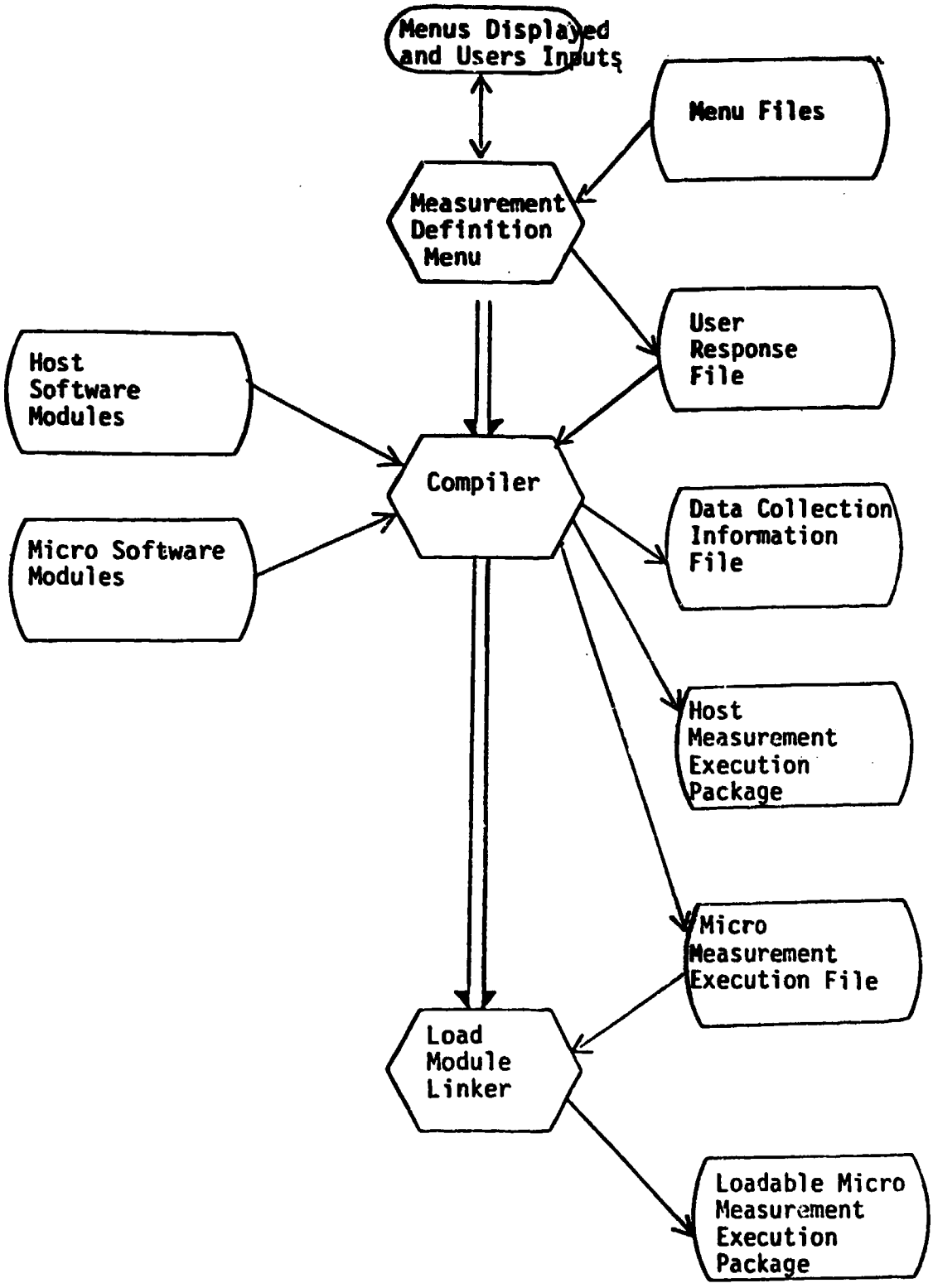


Figure 1-5 Pre-Measurement Data Flow

1.3.2 Measurement Phase

The measurement phase involves the loading of the compiled output, and the execution of the monitor session itself. Figure 1-6 depicts the data flow involved with loading the measurement software into the host and microprocessors. Figure 1-7 depicts the layout of the measurement software once loaded, and the interaction between the components.

1.3.3 Data Collection Phase

Upon completion of a monitor session, the captured data is collected into a data file. This data file is then used by the report generation software to generate the requested reports. Figure 1-8 depicts the data flow for the data collection phase.

1.4 Scenario of IPPA Utilization

The IPPA hardware and software design is oriented to two levels of user. The first level applies to design engineers and makes available the full power of the IPPA system through manipulation of microprocessor and host level software. This level will not normally be used in the field. The second level applies to the system user and performance assessment engineer. It makes available a precision tool kit that provides for the definition of measurement sets and post measurement analysis without the need for detailed awareness of the IPPA logic itself.

The following scenario portrays the sequence of events as seen by a system user in defining and executing a desired set of performance assessment measurements.

Many of the individual measurements identified in Section 1.2 are directly accessible to the system user. This access is provided by a

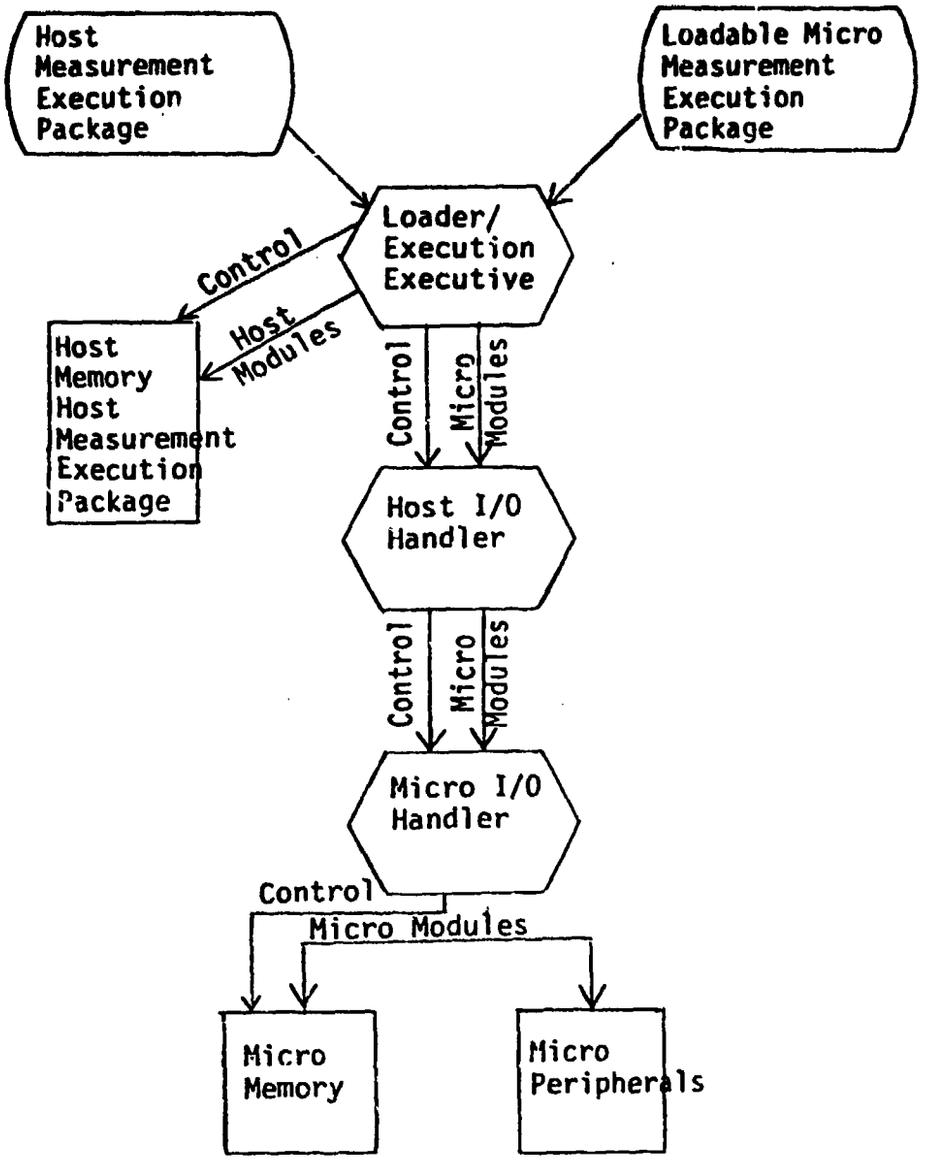


Figure 1-6 Load Data Flow

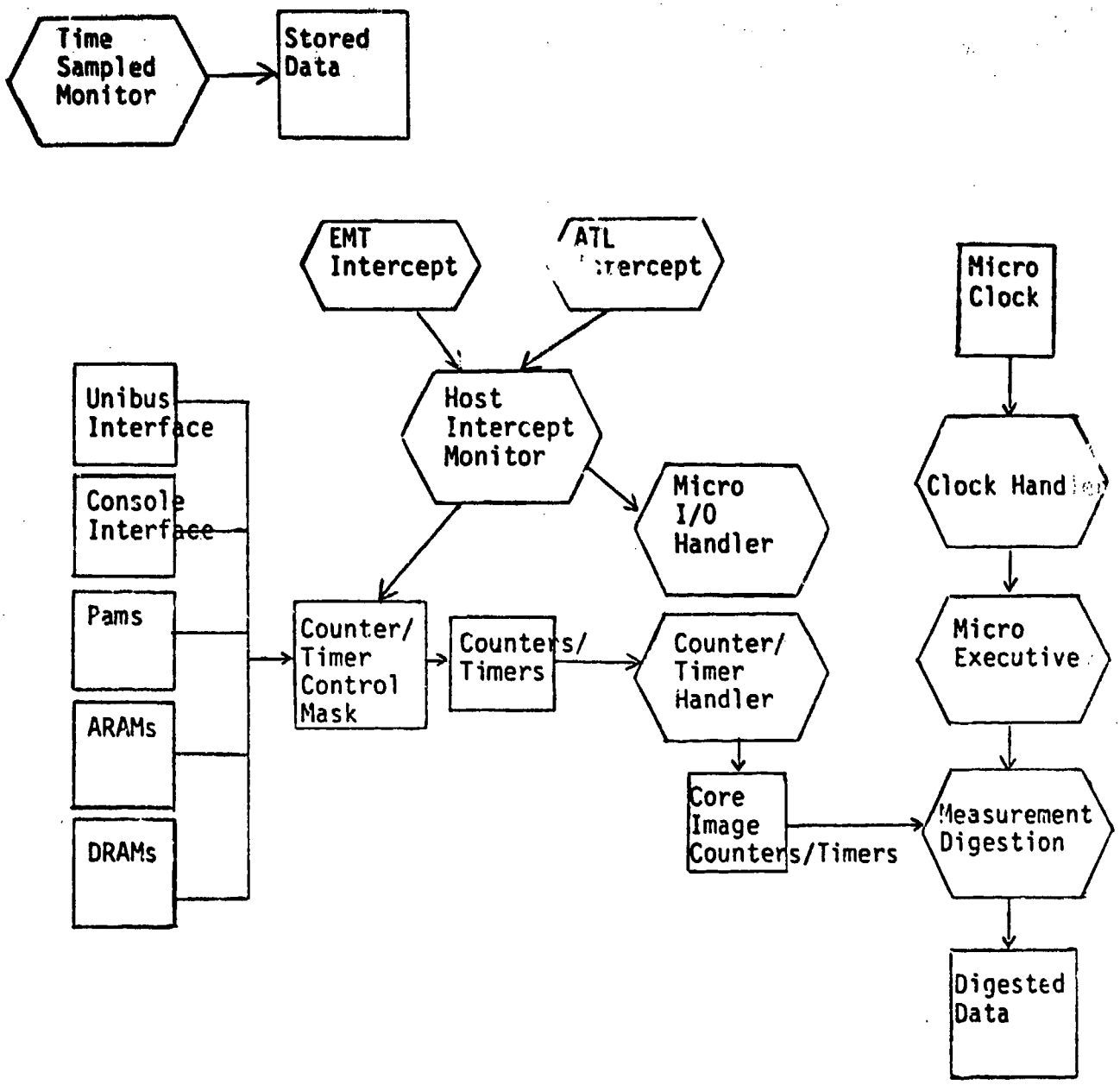


Figure 1-7 Measurement Execution

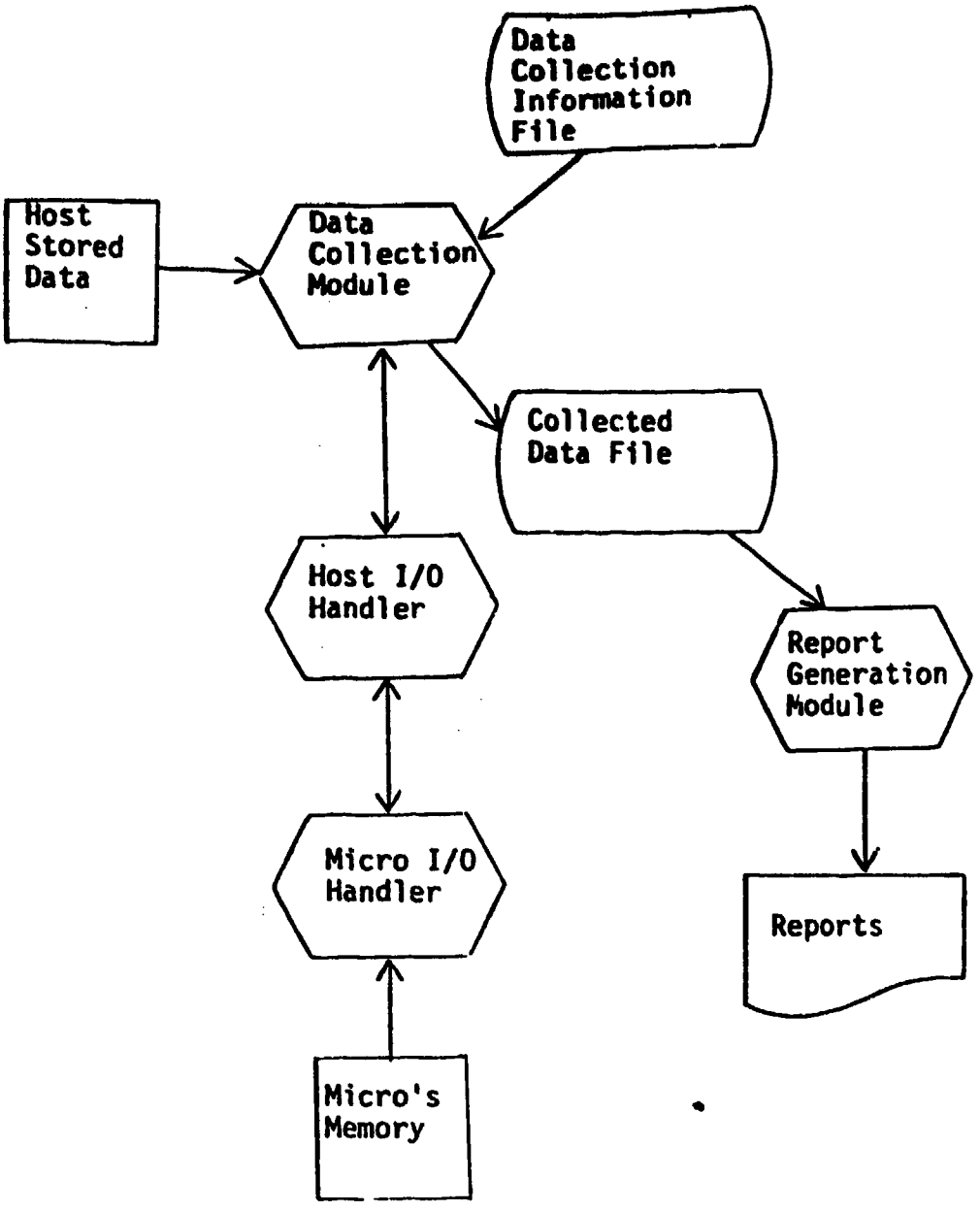


Figure 1-8 Data Collection

command language which enables the user to specify the measurement type and any parameters associated with that measurement. As an example, the detection of task activation and deactivation requires the parameter "Task Name".

Through use of an interactive menu display, the user brings together several such primitive measurements through logical constructs such as "and", "or", or "not". The combination of these primitives is a complex measurement which can be given a specific name and may be stored on disk for later use.

The user will identify one or more complex measurements that are to be concurrently executed on the IPPA. He will then "compile" this measurement set into an executable module - also given a name - that can be loaded into the IPPA upon command. The compiler will collect all necessary software elements and create appropriate table entries for use by the IPPA software. The compiler will also inform the user if the measurement set requires more resources (counters, timers, associative memory loations, etc.) than are available within the IPPA.

The compiled measurement set is also stored on the system disk. It is, however, possible that in later versions much of this information could be stored on peripherals directly connected to the IPPA hardware.

The user may then call for the loading of the measurement set by name. The measurement set can be loaded at a specific future time of day, the same time or times of day for future days, or for one time only.

Once loaded, the software and table structures of the measurement set are in place, and the measurement may be performed.

During a measurement, occasional transfers of information from the microprocessor to the host disk are allowed. This could occur when a "snapshot" function is declared so that statistics over a long period of time can be accumulated and reported for smaller intervals within that time period.

The measurement will complete either upon user command or according to a predefined total time of execution. Upon completion, the last measurements still stored in the microprocessor are transferred to the host disk for storage and later analysis.

Should future versions of the IPPA provide their own bulk memory peripherals, storage of snapshot and measurement completion information could be accommodated on these peripherals with no impact on the host disk system.

Information generated by the measurement compiler is then used by report programs to produce the desired analytical reports on the host system printer.

2.0 SOFTWARE OVERVIEW

This Section contains an overview of the software required for the hybrid monitor. The modules have been divided into four functional areas:

- o Definition and Compilation
- o Host/Microprocessor Coordination
- o Measurement Software
- o Reporting Software

Section 2.1 describes the software modules which interact with the user during the measurement definition phase of a monitor period. This process involves interactive menus which help the user to define the measurements to be collected. Results of this interaction are passed to the compiler, which produces the tables that control the measurement collection process.

Section 2.2 describes the software modules that permit the host computer and the microprocessor to communicate. Communications falls into two categories. The first category of communications concerns transmission of data involved with down-line loading the microprocessor with its controlling software and processing tables, and up-line loading of collected data. The second category of communications concerns raw software measurement data collected by the host software and then transmitted to the microprocessor for processing.

Section 2.3 describes the modules which collect, process and store the measurement data. The processing and storage modules reside on the

microprocessor, while the collection modules reside on both the host computer and the microprocessor.

Section 2.4 describes the modules which create reports using the data collected during a monitoring period.

2.1 Measurement Definition/Compilation

This set of modules interacts with the user to define a measurement session and to create control structures which initialize measurement hardware, perform measurements, process the measurements and report on the data collected. The measurement definition/compilation modules are discussed in the remainder of this section, and have been divided into two groups for discussion purposes:

- Measurement Definition
- Compilation Process

2.1.1 Measurement Definition

This set of modules is responsible for determining the requirements of a user's measurement session.

The measurement definition process also receives inputs from the textual menus, as well as from previously defined measurement sessions stored on disk. The combined inputs to the defintion process are converted into data structures used by the compiler. The definition process also verifies that the measurements requested by the user can be performed and have been properly defined.

The outputs from the definition process are stored on disk and displayed on the user's terminal. The disk structures are divided

into two areas of concern. The first area deals with the static measurements, as well as all the hardware measurements and those software measurements that do not require complex definitions. The second area consists of those software measurements that require complex definitions.

The six definition modules are described in the remainder of this section:

- o User Interaction Module
- o Static Initialization Module
- o Dynamic Initialization Module
- o Monitor Setup Module
- o Help Module
- o Measurement Manipulation Module

2.1.1.1 User interaction Module

This module controls the execution of the entire definition phase of a monitoring session.

- o Inputs - The User Interaction module reads the menu file and the help file depending on the user's input. It will also, when requested, read in the contents of a previous measurement file.

- o Process - This module validates (syntactically) the user's inputs to a menu. On an error it will display an error text

and ask for correct input. When correct data has been entered, this module will perform the indicated function. The functions fall into six categories: help functions, static measurements, dynamic measurements, monitor setup, measurement manipulation and display a new menu.

- o Outputs - This module is responsible for displaying all menus, help texts and error messages to the user. It also passes all user input to the lower level modules to perform the functions requested.

2.1.1.2 Static Initialization Module

This module involves the static measurements. These measurements consist of all hardware measurements and a standard set of software measurements. Three measurement oriented modules are called by this module.

- o Inputs - This module receives data indicating which measurements to perform from the user via the user interaction module. It also reads in formatted tables from disk indicating how to perform the measurement.

- o Process - One of the following modules is called, depending if the measurement is hardware or software related.

- o Outputs - None.

2.1.1.2.1 Hardware Initialization Module

This module translates the user's request for hardware measurements into the table or tables necessary to initialize, process, and report the measurement.

- Inputs - The code number for the measurement requested, the code of how to process the data collected, and an indication as to whether the hardware interrupts on overflow or every time the event occurs.

- Process - Fills the Hardware Static Measurement table with the preload value for the counter/timer, and an indication of how to process the interrupt generated by the counter/timer.

- Outputs - The output consists of a Hardware Static Measurement table necessary to initialize a counter/timer.

2.1.1.2.2 Peripheral Activity Module

This module allows user to monitor a peripheral on the host.

- Inputs - ASCII text from the user interactive module containing the device to monitor and the options chosen.

- Process - The device name is verified to assure the device is configured on the system. The options are then verified to assure they are correct for the device monitored.

- Outputs - The ASCII string received as input will be the output.

2.1.1.2.3 Host Software Module

This module performs an analogous function to the hardware initialization.

- Inputs - This module is supplied with information about the software functions selected. This consists of the code number

of the function, the format to process the data into, and information on the fequency and number of times to collect the data.

- Process - A Software Static Measurement table entry is created that informs the monitor executive about the monitored function. It takes the user supplied inputs and places them in their correct positions in this table.

- Outputs - The output is the Software Static Measurement table that will inform the monitor executive which static software measurements are being performed.

2.1.1.3 Dynamic Initialization Modules

This collection of modules build and validate the data structures necessary to recognize complex software "happenings" and process them. These "happenings" fall into two categories: events and statii. An event is defined as an occurrence. It has no time dimension. Examples of events are the occurrence of an interrupt, the change in a signal level, and the issue of an EMT. Events can be counted or histogramed. A status (plural statii) is defined as a condition. It has a time dimension and can therefore be timed. The change of status is an event. The status "EMT in process" is defined by its starting event (issue of an EMT) and its terminating event (ATL scan, ISR beginning, etc.)

This formality of definition supports experiment definition error detection as, for example, the logical "AND" of an event and a status is equivalent to a "gated" event, the "OR" of two events is always an event, etc.

This collection of modules receives its inputs from the User Interaction module. These modules process the user's input into data structures the compiler uses to set up for complex software measurements. The measurements are also validated to assure that too many measurements have not been defined and that the measurements defined are performable.

The outputs from this collection of modules are: an ASCII table of defined events and an ASCII table of defined statii; an ASCII table of Boolean logic to perform on events and statii and what measurements will be performed after the Boolean has been applied.

2.1.1.3.1 Define Module

The Define module allows a user to define the basic software events to be measured. These events are the basic building blocks of the dynamic measurements.

- o Inputs - This module receives an ASCII string from the User Interaction module. This string can define an event to be in any of the following areas: task execution, EMT execution or the setup data for an ARAM, DRAM or CRAM measurement.

- o Process - The input to the Define module is validated for correct alphanumeric usage and the syntax of the command. If these checks are satisfied the input is passed to the Examine Dynamic Measurement Modules (2.1.1.3.4).

- o Outputs - One output from Define Module is the validated input string. The other ouputs consist of error texts to the user.

2.1.1.3.2 Event Module

The Event Module has two functions; it allows for the logical

combination of previously defined statii and events into new events, and it instructs IPPA to measure an event.

- o Inputs - The input contains the event name to process, how the event is being defined and how to process the event if applicable.

- o Process - This module validates the syntax of the event and the alphanumeric string. The string, if valid, is then passed to the Examine Dynamic Measurement Module (2.1.1.3.4). If not valid the user is told of the error and asked to correct it.

- o Outputs - One of the outputs of this module is the validated input string. The other outputs consist of error texts sent to the users.

2.1.1.3.3 Status Module

The Status module is used to build complex measurements using previously described measurements. The format is a status name, starting event, ending event and optionally an action or status name, Booleian expression, status name and optionally an action.

- o Inputs - The input is an alphanumeric string received from the User Interaction module.

- o Process - This module validates the input received from the User Interaction module. It is validated for syntax and alphanumeric characters. If the input is valid the Examine Dynamic Measurement module (2.1.1.3.4) receives it, otherwise the user is informed of his error and asked to collect it.

- o Outputs - None.

2.1.1.3.4 Examine Dynamic Measurement Module

This module verifies the user's requests for measurements. These verifications include whether sufficient software and hardware support are available, and that all the event and status names used have been defined. The second step of this process is to translate the user's input into the data structures required by the compiler. The Validate Dynamic Measurement and Build Measurement Structure Modules are used by this module.

- o Inputs - The ASCII strings validated by Define, Event and Status modules.

- o Process - The Validate Dynamic Measurement Module is called first. If no validation errors occur then the Build Measurement Structure Module is called.

- o Outputs - None.

2.1.1.3.4.1 Validate Dynamic Measurement Module

This module validates that a user's measurement can be provided.

- o Inputs - The input to this module consists of a verified (syntactically and alphanumerically) Define, Event or Status string, that the user typed in.

- o Process - This module verifies that all event and status names used have been defined. It also checks to see that statii are defined by statii events and that events are defined by an event, or a status and

an event. If the input passes these tests, a success code is returned to the Examine Dynamic Measurement Module. If it does not pass, a failure code is returned to the Dynamic Measurement Module.

- o Outputs - There are two outputs from this module. The first is the success/failure code. The second are any error texts that are displayed to the user.

2.1.1.3.4.2 Build Measurement Structure Module

This module adds newly defined events and statii to the data structures needed by the compiler and the Validate Dynamic Measurement Module.

- o Inputs - The input to this module is the completely validated event or status definition.

- o Process - This module attempts to add the new event or status to the Event or Status Tables. If it can allocate the table space, it then adjusts the resource available counters for each resource necessary for the measurement.

- o Outputs - This module has a success/failure code that is returned to the Examine Dynamic Measurement module. Any errors found in allocating table space or resources available will produce an error text for the user.

2.1.1.4 Monitor Setup Module

This module controls the execution of the monitor run. It can be used

to define the start/stop time for the monitoring session, the frequency of data collection from the microprocessor and host monitor, and a unique filename to store the collected data and run files on. This module calls the Start/Stop, File Name, and Collect setup modules.

- o Inputs - This module receives input from the User Interaction Module.

- o This module validates the syntax and alphanumeric content of the input. If valid, the appropriate setup module is called and the user's input is passed to it.

- o Outputs - The valid user request is passed to the appropriate setup module. If any errors are detected in the statement, an error text is displayed on the user's screen.

2.1.1.4.1 Start/Stop Module

The Start/Stop Module is used to define the starting and/or stopping time for a monitoring session.

- o Input - An input is received from the Monitor Setup Module. A success/failure code is also received from the monitor executive.

- o Process - The start and/or stop time is validated for clock times. If they are valid, a message is sent to the monitor executive. If the monitor executive returns a success code, the measurement session is queued up. If a success code was not received, the user must specify a different start time.

- Output - The message to inform the monitor executive when to start the monitor session, and any error texts sent to the user.

2.1.1.4.2 Filename Module

This module validates a user's filename.

- Inputs - The input text received from the user and a success/failure code from the monitor executive.

- Process - The filename is validated to assure that it is in the format for the host system's file structure. If the user does not supply a name, a default name will be used. This information is sent to the monitor executive. It is also used by the Store directive.

- Outputs - The message sent to the monitor executive to give it the filename, and any error texts sent to the user.

2.1.1.4.3 Collect Module

This module defines the period during which to collect data from the microprocessor and the host's software monitor.

- Inputs - This module receives its input from the Monitor Setup Module and from the monitor executive.

- Process - The input is checked to determine if it is numeric. If so, the period of data collection is sent to the monitor executive.

- Outputs - The message to the monitor executive, containing the period of data collection, and the error texts sent to the user.

2.1.1.5 Help Module

This module will give the user an explanation of any specified area of measurement definition.

- Inputs - The user will input a code through the User Interaction Module.

- Process - Using the code entered by the user, the Help Module will extract a Help text from its text areas. It will then format the text for display on the user's screen.

- Output - The Help text the user has requested will be displayed on their screen.

2.1.1.6 Measurement Manipulation Module

This module facilitates the storage and retrieval of defined measurement sessions. The Delete, Store, Include, and List modules are called by the module.

- Inputs - The inputs to this module are from three sources: the command the user has typed in, the measurement session the user has defined, and the existing measurement sessions.

- Process - This module validates that the command received from the user is correct. It then calls the

appropriate module, or displays an error text at the user's terminal.

- Outputs - Outputs from this module consist of error texts and measurement files.

2.1.1.6.1 Delete Module

This module deletes the specified measurement file from the system.

- Inputs - The input consists of a filename to be removed.

- Process - The file specified is removed from the file system. Should the filename not exist or the user not have deletion privilege, the user will be informed by an error message.

- Outputs - An error message that will tell the user he tried to delete a non-existent file or one that he does not have deletion privileges to.

2.1.1.6.2 Store Module

This module is used to add a new measurement session file to the file system.

- Inputs - The filename of the new measurement session.

- Process - The file system first checks to see if the filename already exists. If so, it informs the user to either delete the filename or to supply a new filename. Then the store module has opened a new

file, the measurement session is written into the file. The file is then closed and it can now be used for a measurement session.

- Output - A new measurement file in the file system.

2.1.1.6.3 Include Module

This module is used to include a previously defined measurement session as part of the current measurement session.

- Inputs - The inputs to this module come from the User Interaction module, and the filename the user specified.

- Process - The filename specified is opened. If the file does not exist, an error text is displayed on the user's terminal. If the file exists, the contents of the file is read and added to the existing measurement set. Any duplicate event or status names are flagged but not added to the measurement set.

- Output - Any error texts caused by the inclusion of a previously defined measurement set.

2.1.1.6.4 List Module

This module has two options; to list the current measurement set, and to list the measurement set specified by the user's input.

- Inputs - The user inputs a filename to be listed or gives no input at all. If a filename has been specified, the file is used as input. Otherwise the current measurement set is used.

- o Process - If a filename has been specified, the file is opened. If the file exists, it is read. The information stored in the file is translated into text to show what measurements are being performed. Then the file is closed. If the current measurement set is being listed, it goes through the same translation process to describe the measurements being taken.

- o Output - A list of measurements to be performed is displayed on the user's screen. Error messages for nonexistent files will also be displayed.

2.1.2 Compilation Process

The compiler takes a measurement session file defined by the user and translates that into the data structure and files necessary to perform the measurement session. This set of modules utilizes the outputs from the Definition set and predefined measurement templates. The templates consist of the data structures for count/timer control and software measurement setup.

The compiler works in two steps. The first step consists of setting up to measure the static measurements. This process consists of building the data structures necessary to obtain the information, process the obtained data, reporting the data and allocating space for the measurement in the microprocessor memory. The second step is to perform the above process on the dynamic measurements.

The Compile process modules fall into three groups, discussed in the remainder of this section:

- o Static Measurement Control
- o Dynamic Measurements
- o Memory Allocation

2.1.2.1 Static Measurement Control

These modules produce all the data structures needed to perform hardware and software static measurements.

The software and hardware static measurement tables produced from a user's measurement session definition and prefilled data tables for the hardware and software static measurement are used by these modules.

The software static measurement tables and the prefilled software static measurement tables are read in, and the software module is called and passed this information. The hardware static measurement tables and the prefilled hardware static measurement tables are then read in, and the hardware module is called and passed this information. The PAM module is also called and passed this information.

2.1.2.1.1 Hardware Module

This module modifies the prefilled hardware static measurement table to reflect the request of the user.

- o Inputs - This module utilizes the hardware static measurement table and the standard prefilled hardware static measurement table.

- Process - For each hardware measurement, this module modifies the prefilled hardware static measurement table producing the Hardware Configuration table. The items modified are arm command, preload count, type of processing to perform and the address to store the processed data. This information is also written to the data description file.

- Outputs - A record in both the data description file and the Hardware Configuration table.

2.1.2.1.2 PAM Control Module

This module builds the data structure to collect and process data from the PAM.

- Inputs - An ASCII string containing device to measure and options to measure.

- Process - The device name is checked with a table of device names, address and measures. The PAM structure is then loaded with CSR's address, CSR bit codes, word count register's address if a disk cylinder address register's address and the cylinder address register's configuration. Then for each option specified that option field in the PAM structure is turned on.

- Outputs - A data structure to control the measurements peformed by the PAM.

2.1.2.1.3 Software Module

This module creates the Time Sampled Software Manager Control table that drives the Time Sampled Software Manager.

- o Inputs - The inputs to this module are the software static measurement table and the prefilled software static measurement table.

- o Process - For each software measurement the prefilled software static measurement table is modified to gather the data the user has requested. The modifications are: the decision to measure or not, and frequency of measurement.

- o Outputs - The Time Sampled Software Manager Control table used by the Time Sampled Software Manager.

2.1.2.2 Dynamic Measurements

These modules translate a user's request for measurements into the data tables necessary to collect, process, and report on the measurements.

The ASCII tables produced by the Measurement Definition Package are utilized by these modules.

Tables that initialize RAM's, install interceptive monitors, process the raw software information, and allow report generation are produced by these modules.

2.1.2.2.1 Host Monitor Module

This module builds a data table that tells the Interactive Monitoring Manager which Interactive Monitoring module to install in the system.

- Inputs - Each "define" statement that the user used in building the measurement session and the software static measurement table is input to this module.

- Process - The software static measurement table is interrogated for measurements that require an interceptive monitor. For each one present in the monitoring session, its entry in the Interceptive Monitor Manager Control table is turned on.

- Outputs - An Interceptive Monitor Manager Control table is produced.

2.1.2.2.2 Associative RAM Setup Module

This module builds the table required to initialize the ARAM, DRAM and CRAM.

- Inputs - The text of define statements referencing ARAM, DRAM or CRAM.

- Process - This module builds the RAM Tables. The information in the ARAM and DRAM statements are: Bit to set, and address range to set it for. This information is put into the ARAM and DRAM Initialization. The CRAM is a bit different. The user specifies which bits he wants to test that are true, which bits to test that are false and which bits

are either condition. The bit to set is stored. Then the bits the user wants to be set are placed in a cleared word that is stored in the CRAM Initialization table. Then the set bits are ended with a word that has all its bits set. Then this word is added to the table entry.

- Outputs - The ARAM and DRAM Initialization and CRAM Initialization tables that will cause the RAMs to be loaded.

2.1.2.2.3 Microprocessor's Dynamic Measurement Table

This module translates the user's dynamic measurements into a series of instructions that will perform them.

- Input - All of the dynamic measurements that the user created during measurement definition.

- Process - Each event name is assigned a number and status measurements a status bit. For each event or status that performs a measurement, the compile gathers all event(s) and status(s) that go into building the status or event. These events and statuses are translated into the instructions required to perform them. Then for each elementary event, an instruction list is produced. This list shows each status or event affected by this event. When every measurement has gone through the above process the instruction lists are ordered and duplicates removed. The order is as followed: set or clear status, process measurements, perform Boolean logic, perform Boolean logic that process measurements, create new events.

This list of instructions becomes the Event Processing table. The address of the starting instruction is stored in the Event Table.

- Outputs - The output is a list of instructions ordered by event number.

2.1.2.2.4 EMT and Task Name Tables Module

This module builds a table that translates measured task names and EMTs into their event numbers.

- Inputs - The inputs are: task name and event number, task name and Mask register bit setting, and EMT number and event number.

- Process - The input is broken into two categories: EMTs and Task names. The EMT number is used as an index into a table. The event number is stored at that position in the table. This creates the EMT conversion table. The Task name is used to search the task name conversion table. If the task name is not in the table, it is added. The event number or Mask register bit setting is added.

- Outputs - A table that converts EMT numbers to event numbers, and a table that converts task names into event numbers and Mask register bit settings.

2.1.2.2.5 Data Description File Module

This module builds a file that describes each and every measurement and where its data is stored in the microprocessor.

- Inputs - The measurement name, type of measurement address the data is stored at and number of elements.

- Process - The data received from every measurement is formatted and written to the Data Description file.

- Outputs - A record in the Data Description file.

2.1.2.3 Memory Allocation Module

This module is responsible for allocation of data storage in the microprocessor.

- Inputs - The number of words of storage needed for a measurement.

- Process - This module takes the current storage address and sends it to the user. It then adds the number of storage words required for a measurement to the current storage address. This will be the next address returned.

- Output - The address that the data will be stored at.

2.2 Host Processor Microprocessor Coordination

2.2.1 Objective

The objective of this software is to establish the communication and functional relationship between the host processor and the microprocessor. As illustrated in Section 3, the microprocessor resides in the midst of the IPPA hardware, and is responsible for controlling much of this hardware. In turn, the microprocessor is controlled by commands from the host processor. This design insures that, ultimately, the user controls the monitor. Two types of communications take place:

- o Communications from the Host Processor
- o Communications from the Microprocessor

2.2.1.1 Communication from the Host Processor

2.2.1.1.1 Measurement data from the Host

The host processor transmits measurement data collected by the software probes and commands to initialize and control the microprocessor - based hardware probes.

At measurement time, the events detected by the software monitor are sent for time stamping, intermediate processing, and storage. This will be discussed in more detail in Section 2.3.1.2.

2.2.1.1.2 Control/transfer commands from the Host

The microprocessor's operational software, as well as driving tables, etc., must be loaded into the microprocessor memory prior to measurement execution time. The actual format for communication of both measurement data and control/transfer commands is quite similar, with the following distinctions:

- Whereas the measurement data directly supports the results to be produced for the output, the control/transfer commands control the microprocessor.

- The measurement data must be serviced, and stored in a minimum of time. The control/transfer commands have some flexibility in execution time.

The control/transfer commands are responsible for setting-up the microprocessor for measurement detection, terminating the measurement detection and transferring the intermediate results to the host processor for report generation (refer to Section 2.2.4).

2.2.1.2 Communication from the Microprocessor

Communication from the microprocessor is accomplished via the device's host interface registers (CSR and FIFO). Utilization of these registers is discussed in 2.2.2 below and in Section 3 - Hardware Design. If error conditions arise, error codes will be put in the status bits of the CSR. Under normal conditions, the microprocessor only communicates the status information to the host. When commanded, however, the microprocessor transmits data stored and collected during measurement sessions. This data is stored on host perpherals for later analysis and reporting.

Depending upon the enable bits of the CSR, the microprocessor has two forms of communication:

- Passive, by just setting the status bits of the CSR

- Active, setting the status and generating an interrupt at the host

The remainder of Section 2.2 will describe the design that supports the host processor/microprocessor communication.

2.2.2 Host Processor/Microprocessor Interface

From the viewpoint of the host, communication with the microprocessor is performed in a manner similar to any standard I/O device. The device control and data registers can be accessed through the peripheral page and consist of:

- A CSR Command and Status Register (CSR)
- A 16 bit mask register (MR).
- A 16 bit data buffer register (DBR).

2.2.2.1 The CSR

This register is the main means of passing control/status information between the host processor and the microprocessor. Status as well as error conditions from the microprocessor are available to the host processor. Also, interrupts can be generated at each processor by setting certain bits in this register.

To send a command from the host to the microprocessor, the host processor places a code in the CSR command field and sets the bit (DEC standard bit 0) of the CSR. This action generates an interrupt at the microprocessor. The microprocessor responds by reading the CSR, and determining from the command field what action has been requested. Setting the "GO" bit clears the "DONE" bit in the CSR. The microprocessor read of the CSR clears the "GO" bit.

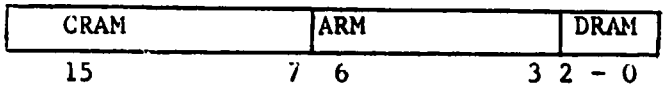
Most of the communications to the host processor are in response to a previously received command.

In response to a host processor command, the microprocessor will usually set a code in the status field representing the microprocessor status. Depending upon the circumstances, the done bit, the error bit, (or no bits) will be set. If it is desirable to interrupt the host processor on one or more of these conditions, a host interrupt request may be generated when the interrupt enable is set.

Detail of CSR bit assignments are provided in Table 3-3.

2.2.2.2 The Mask Register

The mask register contents are used to directly gate (AND) the output of the IPPA's associative RAM's. In this way, each distinct output of the event-detecting RAM may be enabled/disabled by setting/clearing a corresponding bit. The set assignments of the MR to the associative RAM's are:



The MR is directly controlled by the host processor.

2.2.2.3 The DBR

The DBR is the major bidirectional data path for host/microprocessor data transfer. It is a 16 bit wide "port" accessed via a single address on the UNIBUS external page.

To facilitate fast I/O without the added hardware necessary for DMA capabilities, a 128 word FIFO, resident on the microprocessor board, is accessed via the DBR. The use of the FIFO releases the host processor routine from having to check after each word to ensure the microprocessor has succeeded in keeping up to the programmed I/O pace. Also, it avoids the alternate method of generating an interrupt after each word has been transferred.

Functionally, the FIFO will be described in the hardware section. Here it is sufficient that the designed 1.4 user transfer rate is capable of keeping pace with the host processor's programmed I/O. Transfer can be bidirectional under microprocessor control, and appears, from a software point, to be a single word buffer. Transparent to the user, each read or write is automatically stored in the buffer.

2.2.3 Host Commands to the Microprocessor

The two major forms of commands from the host processor are control/transfer commands and measurement data. All command structures are implemented in software/firmware on the microprocessor and are therefore amenable to change, should it be necessary.

2.2.3.1 Control/Transfer Commands

These commands are issued by the host processor, via user interaction, to control the microprocessor and hardware portion of the monitor. This concept allows flexibility in monitor usage, the things to measure, as well as the times to start, stop, and return to user control. Each CSR command can be qualified by a word inthe DBR to extend the initial 16 commands, when necessary. The (CSR) control/transfer commands will include the following:

- Reset command - This command is used to clear portions of memory, and also to reset counters and timers. To allow flexibility in clearing certain portions of memory, the reset command will be qualified to include:

- reset all memory
- reset data tables
- reset control tables
- reset PAM instructions

o WRITE (Down-Line Load) - This command is used to transfer a group of 16 bit words from the host processor to the microprocessor. These words can be microprocessor instructions, data, control tables, etc. The WRITE command will be qualified to include:

  - down-line load executable instructions
  - down-line load control tables
  - down-line PAM instructions
  - down-line load ARAM, CRAM, DRAM values

o Start monitor - After all the information is loaded into memory necessary to complete the measurement requirement, the start monitor command is given. This command arms the required counters/timers, enables the PAM, and Unibus INTR detection circuitry. This would also include any synchronization to wall clock time that is required.

o Stop monitor - The stop monitor command, signals a termination of data collection. All the timers/counters are disarmed, the PAM, and INTR circuitry are disabled. Basically, any devices that were enabled in the start command will be disabled in the stop monitor command- this to insure a uniform measurement. There could still be information in the measurement queue, the microprocessor would be allowed to run in order to empty the queue and complete post collection processing. Also, the counters/timers

could contain information (any value less than the overflow value) which would be accessed and stored in memory.

- READ (Upline Load) - This command can transfer any or all of memory, (as well as PAM memory) to the host processor. In many cases, this command would merely be used to gather the results of the monitor session. The different kinds of information that could be transferred are:

  - upline-load data
  - upline-load error tables
  - upline-load all of memory (memory dump)

2.2.3.2 Measurement Data

This particular command is very similar to control/transfer commands. Both use the CSR to initiate a microprocessor interrupt service routine. The unique types of measurement data are determined (if necessary) by codes placed in the DBR (data buffer register).

2.2.4 Support Routines

Host processor/microprocessor communications cannot be discussed without understanding the environment that will exist within the microprocessor. As with most processor units, the instructions to be performed must be in memory before execution begins. Since the host-microprocessor communication is the only means of transfer (including instructions), the microprocessor must be a "self-boot" system, requiring no operator intervention. Therefore, some routines must exist in non-volatile memory (ROM). At a minimum the following routines must reside in ROM.

- o Initialization procedures - Among other things, it provides the mechanism for host-microprocessor communication

- o Down-line load procedures - provides the means to accept both data and instructions from the host

2.2.4.1 Initialization

The initialization procedure will be the first routine to be executed each time power is applied to the system, by placing a 4-word program status at location 0000 (SEGMENT 0 OFFSET 0). Since the initialization routine is the first routine invoked, it must insure that a means of communication is developed with the host processor. The initialization procedure will perform the following:

- Determine if the interface lines are up and accessible. These would include the Unibus interface, the maintenance interface and the front panel interface.

- Make appropriate status checks of the microprocessor system

- If the power-up does not disarm all counters/timers and PAM, then provide the means to disarm them

- Develop an initial MISR Jump Table. The development of an interrupt is a hardware function, under the control of the "interrupt controller". When an interrupt actually occurs the address of the MISR to be initiated is found in a specific location. These locations must be defined with an address before an interrupt occurs. (This MISR jump table could exist in ROM, but then would not be altered under software control.) The initialization routine will develop these addresses in RAM (read and write memory), where many of these addresses would point back to routines in ROM. Therefore, it can be seen that these addresses could be altered later to point to new or different modules. Some routines that will be resident in RAM will not yet exist. These addresses will point to a general error routine in ROM.

- Set the FIFO direction - As alluded to before, the FIFO can transfer information in both directions. One of the initial multi-word transfers is a down-line load. In order to accept this initial transfer, the FIFO must be set to accept data from the host processor. The FIFO is software controlled by the

microprocessor; therefore the direction, as well as FIFO control registers, are the microprocessor's responsibility (refer to the hardware discussion for a more detailed discussion of the FIFO).

- Initialize the clock - The system clock used by the microprocessor for "time stamping" events must be running when the measurement session begins. The start monitor command will synchronize any time of day relationships. This procedure will insure the system clock is working.

- Enable the "Interrupt Controller" - The interrupt controller is responsible for determining the highest priority among contending interrupt requests. When an interrupt occurs, a vector associated with the interrupt is sent. This vector is stored within the interrupt controller, and is developed initially under microprocessor control. Therefore, during initialization the microprocessor must provide these vectors to the interrupt controller. In the initial state some of these vectors are not known. They must be added at a later time (after down-line load).

2.2.4.2 Down-line Load

This routine must reside in non-volatile memory and will be invoked by a host processor command.

2.2.4.3 Error Routines

Several routines that will service host control commands will reside in non-volatile memory. The error routines involved with the boot process must also reside in non-volatile memory. Refer to the appropriate tables. In addition, certain modules needed to support the design will be discussed. Refer to Figure 2-1.

Hardware section for complete list of error types. The two types of error routines are:

- o Software errors - These types of errors include unacceptable, unexecutable host commands; microprocessor software errors; communication faults. Basically, this would include errors or traps that cannot be resolved.

- o Hardware errors - When certain conditions in the hardware occur, a fatal error condition could exist. In this event the hardware would interrupt the microprocessor, invoking the hardware error routine.

2.3 MEASUREMENT

The measurement data sources are the host software probes and hardware detection. Both will be discussed within this section in terms of the microprocessor software that is responsible for the data's intermediate processing, time stamping and storage.

This section will be concerned with the microprocessor's responsibilities in relation to the measurement data. Section 2.3.3 will give a more detailed discussion on how the software probes detect certain events. The hardware detection circuitry itself is described in Sections 3 and 4.

The microprocessor software uses structured concepts, and revolves around two (2) major areas of responsibility: the microprocessor interrupt service modules (MISR) which are designed to store the data in a circular queue and which use a minimum of processing time; and the microprocessor post-processing module, which takes the data off the queue, performs intermediate processing, and stores it in the microprocessor storage area.

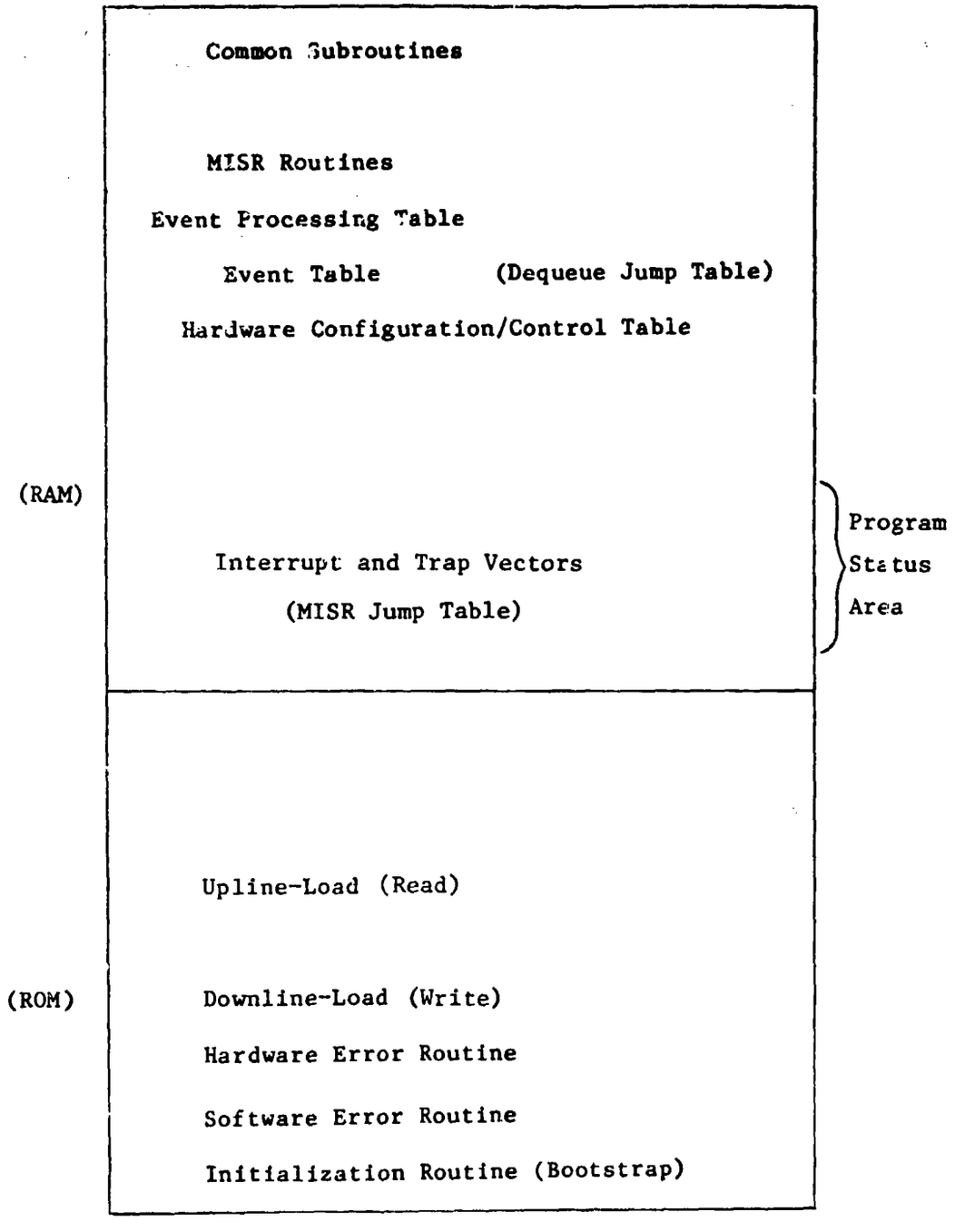


Figure 2-1 Microprocessor Memory Map

2.3.1 MISR Processing

Viewed from the microprocessor, there are several "devices" requiring service under interrupt control. A priority scheme has been developed which allows those "devices" with the greatest need for service to be handled first. MISR processing will be concerned with:

- o Interrupts from the host Unibus interface
- o Command instructions from host processor
- o PAM and VECR function requests
- o CRAM outputs - (via dedicated counters)
- o Counter/Timer overflows

  NOTE: Some ARAM, DRAMS outputs are routed to the CRAM, therefore, these outputs will be discussed in relation to the CRAM.

Each "device" requiring service from the microprocessor generates an interrupt request. The "interrupt controller" determines the highest priority among contending requests and generates a vectored interrupt to the microprocessor. In this way, a unique microprocessor interrupt-service module (MISR) for each device can be initiated.

Each MISR is designed to use a minimum of execution time, to insure contending requests can be serviced quickly with no loss of data.

Also, to miminize context switching time, each MISR will only require the use of a minimum number of registers. The minimum requirements for each MISR are:

- o Disable interrupts
- o Indicate priority level (this goes to the interrupt controller)
- o Save previous data (context switch)
- o Get the Data
- o Store the data
- o Enable interrupts
- o Return from interrupt (and restore context)

The MISR will be responsible for saving enough information to allow follow-on processing to evaluate what actually took place. In many cases this data will be stored in a circular queue. In those cases where data is stored in the queue, a code is developed indicating to the post-processor what the data source was.

The MISR routines will be dedicated to the following measurement devices:

Highest priority

Unibus INTR (VECR)
Command Instructions from Host (CSR)
System Clock
PAM Request
CRAM Counter Outputs
STC Counter/Timer Overflows

Lowest priority

Additional details the assignment of priorities is presented in figure 3-7.

The Z8000 microprocessor can accept up to 256 vectored interrupts. The IPPA design will incorporate 40 unique vectored locations. As each interrupt request is developed by the device requiring service, the interrupt controller will compare the priority of the request with the software priority currently in control. If the requesting device has a higher priority, the interrupt controller will generate an interrupt to the microprocessor.

The interrupt, along with a vector on the address line, will store the previous PC and PSW (flags and status word) on the system stack, and get the new PC from the program status area. This effectively transfers control to the appropriate MISR. The MISR is responsible for saving the contents of any register that it will be using, as well as gathering and storing the data the device has provided.

Each system timer/counter (STC) will have at least 1 MISR to service its overflow. Some STC's will have different MISRs for different functions. In these cases, the vector for that STC will be loaded to point to a particular MISR.

Each of the MISR groups will be discussed in more detail in the following subsections:

- UNIBUS INTR MISR
- Host Command MISR
- PAM MISR
- CRAM (ARAM, DRAM) MISR
- Counter/Timer MISRs

2.3.1.1 Unibus INTR MISR

An interrupt on the Unibus represents a change in state on the PDP-11 system. The times these INTR signals occur help define the state of the system in relation to both hardware and software conditions. Dedicated IPPA circuitry will monitor the INTR signal; when it detects an interrupt it takes the vector associated with the INTR and places it in a 16-word silo. The circuitry then generates an interrupt request to the microprocessor. After the INTR MISR is invoked, it does the following:

- Inhibit interrupts
  - Save registers (context switch)
  - Get time word
  - Get vector from silo
  - Store vector on queue
  - Store time word on queue
  - Restore registers
  - Enable interrupts
  - Return from interrupt

Data will be stored on the queue by a common queue manager routine which controls the pointers, queue size, etc.

2.3.1.2 Host Command MISR

All commands from the host processor invoke the host command MISR. The MISR reads the microprocessor's CSR and determines the type of command. The two major types are:

- Control/transfer commands - These commands control the usage of the monitor.

- Measurement data - This command indicates that some detected measurement data is being transferred from the host processor. This data must be stored on the circular queue for later processing. If it is a multi-word transfer, the first word of the transfer will indicate the number of words to be transferred via the FIFO.

2.3.1.3 PAM MISR

The PAM (peripheral activity module) was developed to monitor device usage; it is an alternative to attaching discrete probes to each device and controller. Because PDP-11 architecture allows software control through access to the peripheral page, device-control signals occur on the Unibus in the form of:

- A unique 18 bit Unibus address describing a particular device register.

- 16 bits of data representing the contents of the device register.

Therefore, for the PAM to monitor Unibus activity without the use of discrete probes, it must be sophisticated enough to accomplish the following:

- Actively monitor the A00-A17 lines, on the Unibus interface in search of from 1 to 4 unique addresses.*

- Upon an address match, transfer the 16 bits of data (D00-D15 the device register contents) to the microprocessor for time stamping and storage.

* This is restrained by the speed of the PAM's microprocessor.

The relatively fast Unibus transfer rate (approximately 1 usec), limits the number of compares the PAM can make. It is not unreasonable to restrict the user to monitoring one device, per PAM, and providing the following possible measurements in relation to this device:

- o NPR's/controller - This measurement is restricted to devices with DMA capabilities.
- o Number and role of interrupt by controller and unit
- o Service time per transfer
- o Disk head position - Obviously restricted to disk units.
- o % peripheral controller busy - For all Unibus devices
- o % MBC busy - for all Massbus devices

To support these Unibus measurements, at least three registers would have to be monitored; they are:

- o The device CSR - The contents of this register will indicate the status of the device; the service, if any, that is required; and error conditions.
- o The word count register (DMA devices only) - This register is loaded with the number of words an NPR device is to transfer. By monitoring this register a good indication of the total of number of transfers, by device, can be accomplished.

- The current head position register (disks only) - This register provides the head position per drive, and in relation to previous head position, it could provide the total number of implied seeks accomplished.

  NOTE: In some cases, the device may have 1 or more registers, in addition to these mentioned, that may have to be monitored. This will depend upon the installation and device types.

The PAM, upon a match, will generate an interrupt request and will transfer at least 2 words to the microprocessor:

- A code representing the PAM function and a code representing the register type (or the lower 11 bits of register address)

- The contents of the device register

The PAM MISR module will be invoked by a vectored interrupt, priorities permitting. This MISR at a minimum will:

- Disable interrupts

- Context switch

- Get a time word

- Put a code representing a PAM function on the queue (this word would include the register type)

- o Put the data on the queue (register contents)
- o Put the time word on the queue
- o Restore registers
- o Return

2.3.1.4 CRAM (ARAM, DRAM)

The CRAM (combinational RAM) is capable of developing some very complex measurements. By inputting some of the ARAM (address RAM) and DRAM (data RAM) outputs, as well as some dedicated hardware lines, several unique combinations of event/event times can be detected. The 16 inputs are:

1. CPU Mode (kernal)
2. CPU Mode (supervisor)
3. CPU Mode (user)
4. CPU Priority Level 7
5. CPU Priority Level 6
6. CPU Priority Level 5
7. CPU Priority Level 4
8. I/O Indicator (direction of transfer)
9. CPU Instruction Fetch
10. OR of MBC and NPR transfer
11. }
12. } ARAM Outputs
13. }
14. }
15. } DRAM Outputs
16. }

At least 4 of the outputs will be sent to 4 dedicated timer/counters. The design will allow either a count of a particular event/combination of events, or a time duration dictated by start/stop events provided by a pair of CRAM output lines.

2.3.1.5 Counter/Timer MISRs

The majority of counter/timers will only require updating a software count of the number of times it has overflowed. In light of this, and the rather slow nature of the overflow rate (a typical rate of 65 millisec), it would seem reasonable to perform the processing within the MISR. This will save on queue size, as well as execution time. Also, as each counter/timer will have a unique vector associated with it, there is no need for polling to determine the particular counter/or timer that overflowed.

The priority of any particular counter/timer is based upon its overflow rate. This rate is dependent upon the input frequency (worst case) and the probability of a continuous input. The 40 counters and timers have been prioritized using this criteria. Table 3-7 gives a list of these counters/timers and estimated priority level.

At initialization time each counter/timer is disarmed. During set-up time those counters/timers to be used will be configured with the necessary status information, and upon a start command from the host, the microprocessor will arm the counter/timers. When a counter/timer overflows, a particular MISR will be invoked. This MISR will perform the following:

- Inhibit Interrupts
- Context switch

- o Establish its software priority level
- o Get a time word (if necessary)
- o Update a software location (dictated by a driving table)
- o Enable interrupts
- o Continue processing if needed (can be interrupted at this point)

  .
  . (Do Histogram, etc.)
  .

- o Restore register contents
- o Return from interrupt

All the counter/timers will store their overflow count in a continuous block of memory. This block of memory would be one of the data tables to be upline-loaded after the measurement is completed.

2.3.2 Microprocessor Post-Processing

To enable the MISR's to respond quickly to service requests, each MISR does a minimum of processing. This places the burden of processing on the post-processing routine. The processes of time correlation, count plus time update, determination of complex events is done by the post-processing routines.

Basically, the post-processors will be executing at the lowest priority level (interrupts enabled) and will be "dequeing" the data placed there by the various MISR's.

The queing scheme envisioned, because of the order of events, has a direct relationship to identifying events. It may, in subtle cases, be more efficient to directly update tables from the MISR.

This queue will contain all the host data, the PAM data, Unibus INTR data, as well as certain outputs of the CRAM counters.

At the "start monitor" command from the host, the post-processing routine will be invoked. The microprocessor post-processing is made up of:

Routines	Tables
Dequeue Manager	Event Table (jump table to a dequeue routine)
Dequeue Routines	Configuration/Control Table (tells the routine what to do)
Common Subroutines	

2.3.2.1 Dequeue Manager

The dequeue manager is responsible for determining which dequeue

routine to invoke. It will take the first word off the queue, use it in the "jump" table to transfer control to a particular dequeue routine. The dequeue manager must also maintain the head pointers, queue size and so on.

2.3.2.2 Dequeue Routines

The dequeue routines have the responsibility to take the information from the queue, perform some intermediate processing, and store the results in reserved memory locations. The dequeue routines are grouped by data sources and are further defined, if necessary, by the information itself. Refer to Figure 2-3 for the queue data sources. The code field, indicating the source, was developed by the particular MISR to provide a means of identifying the dequeue routine required to process the data. Each of the dequeue modules will be discussed with its related data source.

- Unibus INTR Dequeue Module
- Host Dequeue Module
- PAM Dequeue Module
- CRAM Dequeue Module
- Time Dequeue Module
- Terminals Processor

2.3.2.2.1 Unibus INTR Dequeue Module

Since a Unibus INTR indicates a change of state in the host processor, several measurements are dependent upon this information. For instance:

- Any task time measurements would be suspended at the occurrence of an INTR. This time would be provided by the time word associated with the INTR.

- A total number of INTRs by device would be determined, and the various totals would be stored in memory. The device determination would be accomplished by comparing the vector to the device.

The actual module would be controlled by a driving table. The vector would be compared to a list of vectors, (as found in the peripheral page); then the count for the number of INTRs of that device is updated. Also the time for each INTR is put in a stack. This way the various dequeue routines can access this inform tion.

2.3.2.2.2 Host Dequeue Module

Basically for each measurement detection, there is a dequeue routine. Since the host MISR merely stored the information on the queue, the particular dequeue routine is determined from the first word of the data. (This was encoded by the software monitor.)

Each dequeue module is dependent upon the type of data encountered. The code is provided by the host software detection. The particular dequeue module, controlled by a unique driving table, will process the data and store it in a reserved memory location.

The actual processing could be as simple as taking the difference in time between two events and adding that to a total time stored in memory.

2.3.2.2.3 PAM Dequeue Modules

Each device monitored could have one or more registers associated with it, and since each device usually has a unique register layout, it would be most convenient to have a dequeue module for each device. The detected Unibus address would correspond to a particular device. This would provide a means to identify the unique layout of a device register, and to determine the event(s) by comparing specific bits of the register.

2.3.2.2.4 CRAM Dequeue Modules

At least four of the outputs of the CRAM will be tied to dedicated counters. These counters can define a time span, per a pair of counters, or provide a total count of events. Each counter will have a dequeue routine.

2.3.2.2.5 Time Dequeue Module

This dequeue module supports the (STC) System Clock used for time stamping. To minimize MISR execution time, a single time word is accessed. In order to distinguish between time words of a long duration, every timer overflow is put on the queue. This defines the time words in relation to the sequence of events.

When this dequeue module is invoked it updates a known memory location, representing the number of times the STC overflowed. In this way, each dequeue module can relate time words of several different events.

2.3.2.2.6 Terminate Processing Module

This dequeue module is invoked at the conclusion of post-processing.

When a stop monitor command was issued, a code was put at the end of the queue. This allowed post-processing to continue, even though all detection was suspended. This module then performs any clean-up or concluding processing required, and notifies the host processor of the post-processing status.

2.3.3 Host Software

This set of modules controls the collection of Host measurements and their transmission to the microprocessor. The input to this module set is in the form of the systems tables lists and parameters of routines. The monitoring of the system falls into two categories: time sampling, and interceptive sampling. The derivation of each measurement requires examining processing overhead incurred for the measurement and how dynamic or static the measurement is. The output from this set of modules is a few word-transfers to the microprocessor.

The modules in this set are subcategorized and discussed as follows:

- o Time Sampled Software Manager
- o Interceptive Monitoring Manager
- o Monitor Executive
- o Trace Trap Monitor
- o Instruction Count Module

2.3.3.1 Time Sampled Software Manager

This module controls a set of modules which perform measurement or

functions that are best serviced by periodic operation. The modules called by this module include the UMR Module, PUD Module, System Module, and Collection Module.

- o Inputs - The Time Sampled Software Manager Control table, indicating which of the modules to execute and how often to execute them.

- o Process - A length of time is computed until the next measurement(s) and or function(s) will be performed. A mark time directive is issued for this period. When the mark time directive expires, the measurement(s) and/or function(s) are resumed. This process continues until the monitoring period is over with.

- o Outputs - None.

2.3.3.1.1 UMR Module

This module counts the number of Unibus Mapping registers in use.

- o Inputs - None.

- o Process -This routine suspends itself, and is resumed by the Time Sampled Software Manager. When resumed it counts the number of UMRs in use, and calls the communications routine to send the data to the microprocessor. This process is repeated for the duration of the measurement session.

- o Outputs - The number of UMRs in use.

2.3.3.1.2 PUD Module

This module collects data on the devices configured in the system, their on-line or off-line status, and their logged-on status.

- o Inputs - None.
- o Process - This module is suspended. When the Time Sampled Software Manager resumes this routine, it scans the Physical Unit Directory. Each device type and its units are examined and data is collected on it. It sends this data to the Communications routine to send it to the microprocessor. This process is repeated until the measurement session ends.
- o Outputs - The data the PUD Module has collected.

2.3.3.1.3 System Module

This module collects a count of the number of active tasks in memory and the number of activation of tasks since the last sample.

- o Inputs - None.
- o Process - The system module suspends itself. When a resume is issued by the Time Sampled Software Manager this routine scans the Active task list. It counts each task in the list that can become active. Then it extracts the activation count from the ATL routine. It zeros the activation count. Then the data is sent to the communications routine to be sent to the microprocessor.
- o Outputs - The count of active task and the number of task executions.

2.3.3.1.4 Collection Module

This module collects all the data stored in the microprocessor's memory.

- o Inputs -The size of the data stored in the microprocessor and the filename to store the information in.

- o Process - This module suspends itself. When resumed by the Time Sampled Software Manager it opens the passed filename. It then queues "reads" to the microprocessor through its handler. When the reads have been received, this routine appends the information to the opened file. When all the data has been read, in an "End of Read" QIO is sent to the MC handler and the file is closed. This process is repeated until the end of a monitoring session.

- o Outputs -The QIO commands to the microprocessor's handler and the file written to disk.

2.3.3.2 Interceptive Monitoring Manager

These modules make best use of obtaining the information as it occurs. This is accomplished by modifying portions of the executive code to obtain the data.

The following modules are called by this module: Node Monitor, Task Activation Monitor, EMT Monitor, Directive End Monitor, Trap Monitor, IOT Monitor, TRAP4 Monitor, Memory Utilization Monitor, Task Execution Monitor, Checkpointing Monitor.

- o Inputs - The Interceptive Monitoring Manager Control table, indicating measurements to perform, is received from the Monitor Executive.

- o Process - Each entry in the table contains the information required to patch the executive, perform the data collection subroutine and restore the system to its original form. This module performs each patch required by the measurement session. At the conclusion of the measurement session this module restores the executive to its original state.

- o Outputs - A success/failure message to the Monitor Executive.

2.3.3.2.1 Node Monitor

This module records all allocations and deallocations of nodes in the host environment.

- o Inputs - Its inputs are the same as the inputs to the system's node allocation and deallocation module.

- o Process - After the Interceptive Monitoring Manager has modified the executive, every request for nodes or returning of nodes to the system goes through the node monitor. It copies the information passed to these routines and sends that information to the communications routine.

- o Outputs - The number of nodes allocated or deallocated is given to the communication routine.

2.3.3.2.2 Task Activation Monitor

This module is used to inform the microprocessor of task activation.

- o Inputs - A table of task names, event number and mask register bit setting for the task.

- o Process -After the interceptive Monitoring Manager has installed this routine, task activations are known. This module is added to the end of the active task list scanner. The next active task is compared against the table of task names, event numbers and mask register setting. If a match is found the mask register is written with the contents for that task name. Then the event number is sent to the communications routine. If no match is found, this routine sends a non-monitored task code to the communications routine to be sent to the microprocessor.

- o Outputs - The mask register bit setting and the event number indicating a monitored task is nominated or not.

2.3.3.2.3 EMT Monitor

This module monitors EMT execution.

- o Inputs - This routine uses the EMT Conversion Table and the event number associated with it.

- o Process - When the Interceptive Monitoring Manager has installed this monitor the issuance of every EMT is

detected. It compares the EMT to the EMT conversion table and sends its corresponding event number to the communications routine.

- o Outputs - The event number associated with the EMT.

2.3.3.2.4 Directive End Monitor

This module is used to time EMT execution.

- o Inputs - None.
- o Process - After being installed by the Interceptive Monitoring Manager the end of each EMT is known. The end of EMT directive event number is sent to the Communications routine.
- o Outputs -The event number of end EMT execution.

2.3.3.2.5 Trap Monitor

This module monitors the trap instruction for the purpose of monitoring task times.

- o Inputs - None
- o Process - After installation of this module by the Interceptive Monitoring it will send the trap event number to the communications module
- o Outputs - The event number associated with the trap instruction.

2.3.3.2.6 IOT Monitor

This module monitors the IOT instruction for the purpose of monitoring task times.

- o Inputs - None.
- o Process - After installation of this module by the Interceptive Monitoring it will send the trap event number to the communications module.
- o Outputs - The even number associated with the trap instruction.

2.3.3.2.7 Trap4 Monitor

This module monitors the Trap4 which is a collection of CPU error conditions. Slave faults, memory out of bounds, odd address, yellow and red stack violations are all indicated through the trap.

- o Inputs - None.
- o Process - After the Interceptive Monitoring Manager has installed this module, it will receive all Trap4 occurrences. This module will read the CPU Error Register. The register contents and a Trap4 event number will be passed to the communications module.
- o Outputs - The CPU error register and the Trap4 event number.

2.3.3.2.8 Memory Utilization Monitor

This module monitors the allocation and deallocation of memory. This occurs when a task is being installed, checkpointed or terminated.

- o Input - None.
- o Process - After the Memory Utilization Monitor has been installed by the Interceptive Monitoring Manager, all allocation/deallocation are measured. This module collects the following information: allocation or deallocation, system or user controlled partition, partition name and the number of 32 word blocks allocated or deallocated. This information is passed to the Communication routine.
- o Output - The allocation/deallocation of memory, partition name, user or system controlled partition and the amount of memory requested.

2.3.3.2.9 Task Execution Monitor

This module detects the return of execution to the previously executing task or to another task.

- o Inputs - None.
- o Process - After the Task Execution Monitor is patched into the Interactive Monitoring Manager, every time the executive tries to return to a task the outcome is known to this routine. The outcome falls into two classes. The first is that control is returning to the task that last had control of the CPU. The other case is when control is returned to someone other than the last user of the CPU. Each case has an event

number and this is passed to the communications module

- Outputs - The event number for a return to previous task or return to new task.

2.3.3.2.10 Checkpointing Monitor

This module receives information on tasks being checkpointed and rolled in.

- Inputs - The information passed to the checkpointing and rolling in routines.

- Process - After the Checkpointing Monitor has been installed by the Interceptive Monitoring Manager all roll-outs and roll-ins of tasks will be recorded. The information collected are the following: roll-in or out, partition name and size of task brought in or sent out. This information and the event number are passed to the communications module.

- Outputs - The information collected on the rolling in and out of tasks.

2.3.3.3 Monitor Executive

This module controls the flow of information and execution to the parts of the monitor.

- Inputs - A user command to run a measurement session from the terminal or the start of measurement session that was defined in the definition software. The success/failure codes received from the down-line

loader, Time Sampled Software Manager and Interceptive Monitoring Manager.

- o Process - This module first starts the down-line loader. It then passes it to the filename for the current session. It then waits for information on the success or failure of the down-line load. If it was a failure, the user is informed and the process is terminated. If successful, the controlling table for the time sampled software is read and passed to the Time Sampled Software Manager. The Monitor executive then waits for the Time Sampled Software Manager's reply. If successful, the Monitor Executive waits for the measurement session to end normally (defined stop time) or the user to terminate it. At this time, the Interceptive Monitoring Manager is informed to restore the system to normal. The collection routine is run and all other software is stopped.

- o Output - All commands and information sent to the down-line loader, Time Sampled Software Manager and the Interceptive Monitoring Manager.

2.3.3.4 Trace Trap Monitoring

This module is not run during a monitoring session.

- o Inputs - None.

- o Process - This module installs itself into the operating system. It counts the number of times the trace trap has been executed. On user requests to this program it will print out a count of the number of trace traps executed.

- Outputs - The count of the number of trace traps.

2.3.3.5 Instruction Count

This module is provided for debugging programs. It will execute the program one instruction at a time and keep track of instructions executed.

- Inputs - The name of the task that will have its instructions counted.

- Process - This module will modify the trace trap vector to point to itself. It will then change the programs processor status word to have the T Bit set. When a trace trap occurs this program determines if the "monitored" task issued it. If it does, it updates its instruction counts and returns control to the program. If not, it allows the normal trace trap program to deal with the trace trap.

- Outputs -The output will consist of a count of the instructions executed.

2.4 Measurement Reporting

This collection of modules displays all the collected data on the user's screen or the system's print device. Appropriate modules read the collected data file, the data description file and the static measurement table. All static measurements are printed in their report formats, followed by the dynamic measurements.

The modules are discussed below in two groups:

- o Static Measurement Reports
- o Dynamic Measurement Reports

2.4.1 Static Measurement Reports Module

This module is used to report the static measurements. These measurements remain the same (if chosen) for every run of the monitor. The following report modules are called by this module as required: Node Report, System Report, Unibus Report, Fast Bus Report, Mass Bus Report, PUD Report, CPU Report, Fault Report.

- o Inputs - This module reads the static measurement table, the data description file and the collected data file.
- o Process - The static measurement table is read. For each measurement taken, the description file is searched for its entry. When found the correct subroutine is invoked and the report for that measurement is printed.
- o Outputs - None.

2.4.1.1 Node Report Module

This module will print the total number of nodes used by the system at each sampling period.

- o Inputs - The location and length of the node data in the collected data file is passed from the Static Reports Module.

- o Process - The node data stored in the collected data file is read in. A node report heading is produced. The sample times and the node counts are printed for each sample.

- o Outputs - The report containing the sample counts and the sampling times is the output.

2.4.1.2 System Report Module

This module prints a report on system performance.

- o Inputs - The location of the system data storage area in the collected data file and the number of entries in the system data storage area.

- o Process - This module prints a heading for the system report. Then for each sample, the time of the sample and the data collected will be printed. The data consists of the number of active tasks in memory, the task request rate and the number of memory block (32 words to a block) - seconds that are available on the system.

- o Outputs - The report on the system performance.

2.4.1.3 Unibus Report Module

This module produces a report on the Unibus. The data contained in the report are Unibus occupancy, Non-Processor Request (NPR) contention, Bus Request (BR) contention at each level, Unibus Mapping Register (UMR) utilization and transfer per second.

- Inputs - The inputs to the Unibus module are the address of the Unibus data in the collected data file.
- Process - The Unibus Report module prints a heading. The data for the Unibus measurements are printed with a title to describe what measurement it is.
- Outputs - The Unibus Report is generated as output.

2.4.1.4 Fast Bus Report Module

This report will only be produced on a PDP 11/70. This is because the PDP 11/45 does not have cache memory. The data contained in this report are cache hit rates, cache/memory transfers per second, read and write counts, and cache contention from the CPU, Unibus and Massbus controller.

- Inputs - The location of the Fast Bus data and its length on the collected data file.
- Process - This module will print the heading for the Fast Bus report. It will read the data from the collected data file, process the data, attach a heading to each data item and print them.

- Outputs - The Fast Bus report will be the output from this module.

2.4.1.5 Mass Bus Report Module

This report is produced for an 11/70 only. It contains transfer rates for each Mass Bus controller.

- Inputs - The address and length of the Mass Bus data in the collected data file.

- Process - The Mass Bus heading is printed, followed by the read and write rate of each Mass Bus controller on the system. This data is read from the collected data file.

- Outputs - The Mass Bus report is the output produced.

2.4.1.6 PUD Report Module

The Physical Unit Device (PUD) report contains information the devices configured to the system, their on-line and log-on status.

- Inputs - The address and length of the PUD data in the collected data file.

- Process - The PUD heading is printed. The PUD data is read in from the collected data file. The data is translated, formatted and printed with an explanatory heading.

- Outputs - The PUD report is produced as output.

2.4.1.7 CPU Report Module

This report gives statistics on the Central Processing Unit (CPU). The data reported on consists of the following: Percent time at kernel supervisor and user mode, percent time at priority levels (4,5,6,7), percent the CPU is not busy, percent CPU-I/O overlap, percent CPU-FFP overlap, count of floating point instructions and their rate.

- o Inputs - The input to the CPU report module is the address of the data on the collected data file.

- o Process - The heading for the CPU report is printed. The CPU data is then read from the collected data file. The percentages are computed and printed with an explanatory text.

- o Outputs - The CPU report is produced as output.

2.4.1.8 Fault Report Module

This report contains data on error conditions. The errors reported on consist of the following: slave faults, unclaimed NPG and BGs, memory of bounds, odd address error and multiple NPR and BR counts.

- o Inputs - The address of the fault data on the collected data file is received as input.

- o Process - The Fault report heading is printed. The fault data is read from the collected data file. This data and explanatory headings are printed.

- o Outputs - The fault report is produced as output.

2.4.2 Dynamic Measurement Report Module

This report contains the data collected for every Dynamic Measurement. Since these measurements may change drastically from one measurement session to the next, no standard report format can be specified. Four modules are called by this module: Heading; Count; Time; Histogram.

- o Inputs - The data description file and the collected data file are used as input.

- o Process - The data description file is read. For every dynamic measurement, the data associated with it is read. The heading module is called to print text describing the measurement. Then one of three measurement modules is called depending on what type of data was collected. The three routines are count, time and histogram.

- o Outputs - The text that describes a measurement and the values read in from the collected data module.

2.4.2.1 Heading Module

This module produces a heading text for every dynamic measurement.

- o Inputs - The input to this module is a text that describes the measurement.

- o Process - This module expands the text into a more human understandable format and prints it.

- o Outputs - The expanded description text for a measurement.

2.4.2.2 Count Module

This module prints the numeric value of a counted measurement.

- Inputs - The data words containing the number of times the measurement occurred.
- Process - The data is converted from a binary number into an ASCII string. The ASCII string is then printed.
- Outputs - The ASCII string corresponding to the binary number.

2.4.2.3 Time Module

This module converts the binary representation of a time into an ASCII string of days, hours, minutes, seconds and fractions of seconds.

- Inputs - The binary data for the time or length of time of a measurement.
- Process - The binary data is divided by the number of fractions in a second. This leaves a number of seconds and the remainder. This remainder is the fractions of seconds. The number of seconds is divided by 60. This gives the number of minutes and the remainder is the number of seconds. The number of minutes is divided by 60 giving hours and minutes as a remainder. The hours are divided by 24 giving days with hours as a remainder. These numbers are translated into ASCII strings and printed.

- Outputs - The day, hour, minute, second and fraction of a second that a measurement took or the time of a measurement.

2.4.2.4 Histogram Module

This module produces a histogram from the data collected for a measurement.

- Inputs - The address of the histogram data in the collected data file and the number of data element.

- Process - The data for the histogram is read in. Each data element is added to a total count. This count is used to scale the output. Then the total count is divided by the number of lines on a page. This number is used to scale the output.

- Outputs - The output is a histogram of the measurement.

- Outputs - The day, hour, minute, second and fraction of a second that a measurement took or the time of a measurement.

2.4.2.4 Histogram Module

This module produces a histogram from the data collected for a measurement.

- Inputs - The address of the histogram data in the collected data file and the number of data element.

- Process - The data for the histogram is read in. Each data element is added to a total count. This count is used to scale the output. Then the total count is divided by the number of lines on a page. This number is used to scale the output.

- Outputs - The output is a histogram of the measurement.

3.0 HARDWARE DESIGN

This section presents details on the design and theory of operation for the microcomputer-based hardware components of the IPPA. Section 3.1 provides the overview of the hardware elements necessary for understanding the detailed theory of operation found in Section 3.2. Supplementary timing data and parts lists are provided in Section 4 of the volume. Manufacturer specifications for components of the design are listed in section 6 of this volume.

3.1 Hardware Overview

Overall operation of the IPPA is controlled by the Microcomputer (MC) and its associated software program. The MC consists of Z8001A microprocessor and its support circuitry, program ROM, program RAM, and various I/O devices. To increase performance the MC utilizes memory-mapped I/O rather than the standard I/O address space, which is otherwise available using a small set of I/O instructions. The following devices occupy the MC I/O space:

(1) Control and Status Register (CSR)

(2) Mask Register

(3) Error Register (ERR)

(4) FIFO

(5) System Timing Controller (STC)

(6) Interrupt Controller (IC)

(7) Vector Register (VECR)

(8) Peripheral Activity Module (PAM)

(9) Serial Communicatons Controller (SCC) - optional.

(10) Associative Memory (AM)

(11) Qualifier Logic (QL)

Data flows into the IPPA from the host processor via the host/microprocessor control interface and the high speed preprocessing circuitry (Qualifier logic) connected to signal sources on the host. Special purpose timing and counting logic (STC, PAM, AM) under MC control handles much of the heavy work and passes the MC only that data (via the IC) necessary for further organization and/or action. The MC is responsible for configuring the special purpose logic to perform the necessary functions and for the accumulation and organization of collected data that will eventually be included in the system reports.

The following sections describe each of the IPPA hardware components according to their function.

3.1.1 Microprocessor Elements

3.1.1.1 Microprocessor

The Z8001A is a 16-bit microprocessor with segmented addressing which runs at a clock frequency of 6.0 Megahertz.

Support circuitry includes a timing generator, status decoder, address latches, and data and control signal buffers. The status, address, data, and control lines make up the Microbus.

3.1.1.2 Microbus Address Space

The Microbus (MB) uses 19 of the 23 available address lines. This yields an address space in the range from 00000H to 7FFFFH. (The 'H' suffix indicates use of a hexadecimal number base). The upper three Microbus address lines are derived from the lower three Segment output signals (i.e., SN0, SN1, SN2), which are generated by the microprocessor. These three address lines are decoded to select one of the eight memory "pages", each containing 64K-bytes. Table 3-1 shows the mapping of the MC address space.

Page	Address (hex)	Utilization
0	00000 - 00FFF	2K Program ROM (minimum size)
	01000 - 07FFF	expansion Program RIM (up to 16 K maximum)
0,1	08000 - 1FFFF	unassigned
2	20000 - 27FFF	16 Program RAM (minimum size)
2,3	28FFF - 3FFFF	expansion Program RAM (64K maximum)
4,5	40000 - 5FFFF	Associative Memory (64K)
6	60000 - 6FFFF	unassigned
7	70000 - 7FFFF	I/O space (see Table 3-2)

Table 3-1 MC Address Space Mapping

Within the I/O space the 16-bit address in partitioned into fields as depicted in Figure 3-1. Table 3-2 lists the specific addresses for the MC I/O devices.

15 11	10 8	7 5	4 0
Non-existent I/O	Device (0-7)	Unit Number (0-7)	Register Address (0-31)

Figure 3-1 MC I/O Address Devices

Address (hex)	Device
00XX	Control and Status Register
01XX	Error Register
02X0 - 02XF	FIFO (as follows)
0200	Control Register 0
0202	Control Register 1
0204	Interrupt Status Register 0
0206	Interrupt Status Register 1
0208	Interrupt Status Register 2
020A	Interrupt Status Register 3
020C	Interrupt Vector Register
020E	Byte Count Register
0210	Byte Count Compare Register
0212	Control Register 2
0214	Control Register 3
0216	Message Out Register
0218	Message In Register
021A	Pattern Match Register
021C	Pattern Mask
021E	Data Buffer Register

Table 3-2 MC I/O Devices Addresses (page 1 of 3)

Address (hex)	Device
0300 - 03FF	System Timing Controllers (as follows)
0300	STC 0 Data Register
0302	STC 0 Control Register
0304 - 031F	Redundant
0320	STC 1 Data Register
0322	STC 1 Control Register
0324 - 033F	Redundant
0340	STC 2 Data Register
0342	STC 2 Control Register
0344 - 035F	Redundant
0360	STC 3 Data Register
0362	STC 3 Control Register
0364 - 037F	Redundant
0380	STC 4 Data Register
0382	STC 4 Control Register
0384 - 039F	Redundant
03A0	STC 5 Data Register
03A2	STC 5 Control Register
03A4 - 03BF	Redundant
03C0	STC 6 Data Register
03C2	STC 6 Control Register
03C4 - 03DF	Redundant
03E0	STC 7 Data Register
03E2	STC 7 Control Register
03E4 - 03FF	Redundant

Table 3-2 MC I/O Devices Addresses (page 2 of 3)

Address (hex)	Device
0400 - 04FF	Interrupt Controllers (as follows)
0400	IC 0 Data Register
0402	IC 0 Control Register
0404 - 041F	Redundant
0420	IC 1 Data Register
0422	IC 1 Control Register
0424 - 043F	Redundant
0440	IC 2 Data Register
0442	IC 2 Control Register
0444 - 045F	Redundant
0460	IC 3 Data Register
0462	IC 3 Control Register
0464 - 047F	Redundant
0480	IC 4 Data Register
0482	IC 4 Control Register
0484 - 049F	Redundant
04A0 - 04FF	IC Expansion
05XX	Vector Register (read only)
0600 - 06FF	Peripheral Activity Module (utilization T.B.D.)
07X0 - 07X7	Serial Communications Controller (utilization T.B.D.)
07X8 - 07XF	Redundant
0800 - FFFF	Non-existent I/O

Table 3-2 MC I/O Devices Addresses (page 3 of 3)

Figures 3-2 and 3-3 show the address decoding for the STC and Interrupt Controller circuits.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	1				X	X	X		0

STC

0 = Data
1=Control

Figure 3-2 STC Address Decoding

0	0	0	0	0	1	0	0		X	X	X	0
								IC				0 = Data 1=Control

Figure 3-3 Interrupt Controller Address

3.1.1.3 Program ROM

The program ROM circuitry has been designed to accommodate from 2K-words to 16K-words of either EPROM, for development purposes, or high-speed PROM. Pin-compatible 2K-by-8 and 4K-by-8 packages have been designated which should offer a high degree of versatility. A patch plug is used to select the proper addressing scheme for the type and quantity of ROMs being used. The patch plug is also used to select the number of any necessary WAIT states which must be generated when the access time of the ROM is greater than approximately 220 nanoseconds. Each WAIT state adds 167 nanoseconds to the memory's permissible access cycle.

3.1.1.4 Program RAM

The Program RAM has also been designed to allow flexibility in RAM usage. Either 16K-by-1 or 64K-by-1 dynamic RAMs may be used, which provides a maximum of 64K-words of read/write memory. A patch plug is used to select the proper addressing scheme for each type of RAM.

Program RAM is accessible on either a word or byte basis, and parity generation and checking is included for both the low and high byte.

3.1.1.5 Refresh

The Z8001A must be programmed to generate the necessary refresh cycles for the dynamic memory. This is done simply by loading the desired value in the Rate Field and setting the Refresh Enable (RE) bit in the Refresh Control Register. The RE bit enables the internal refresh mechanism, while the Rate Field determines the period between refresh cycles. If 'n' represents the value placed in the Rate Field and ZCLK is the Microprocessor clock frequency, then the refresh period (RP) can be found using the formula:

$$RP =$$

or

$$n =$$

The Program RAM, as well as the dynamic RAM used in the Associative Memory (Paragraph 3.1.3), requires a refresh period not to exceed 15.625 microseconds. Thus,

$$n =$$

$$n = 23.4375$$

The value placed in the Rate Field must not exceed 23 (and must be greater than 0) to ensure proper RAM operation.

3.1.2 Microcomputer I/O Devices

The devices which respond to the MC I/O address space are:

(1) Host Interface Registers (Section 3.1.2)

(2) Error Register (Section 3.1.2.2)

(3) System Timing Controllers (Section 3.1.2.3)

(4) Interrupt Controllers (Section 3.1.2.4)

(5) Vector Register (Section 3.1.2.5)

(6) Peripheral Activity Module (Section 3.1.2.6)

(7) Serial Communications Controller (Section 3.1.2.7) (optional)

The I/O address space is described in Paragraph 3.1.1.2.

3.1.2.1 Host Interface

The IPPA acts a "dumb" peripheral device in the host PDP system. This means that all interaction between the host computer and the IPPA is initiated and controlled by the host.

The IPPA responds directly to a block of four word-addresses in the PDP Peripheral Page. This block of addresses may be located anywhere in the Peripheral Page by appropriately setting the Address Select DIP switches. These addresses (octal) access the following registers:

- Control and Status Register (address 7XXXX0)
- Mask Register (address 7XXXX2)
- FIFO Data Buffer Register (address 7XXXX4)
- FIFO Control and Status Register (address 7XXXX6)

All registers are read/write with the exception of the upper byte of the CSR, which is read only.

3.1.2.1.1 Control and Status Register (CSR)

The bit assignments for the CSR are shown in Figure 3-4.

DONE	IDE	IEE	CMD3	CMD2	CMD1	CMD0	GO
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

CSR - lower byte

Note: denotes read only

HARD ERROR	SOFT ERROR	PROGRAM ERROR	FIFO ERROR	MEMORY ERROR	TIMEOUT ERROR	MICRO	RUN
bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8

CSR - upper byte

Figure 3-4 Bit Assignments

The CSR bit assignments are defined in Table 3-3.

BIT	FUNCTION
0	The 'GO bit initiates the command defined by CSR bits 1-4.
1-4	'CMDO - CMD3' define the command issued to the IPPA (see Table 3-4).
5	Tne 'IEE' bit (Interrupt on Error Enable), when set, allows the IPPA to generate an interrupt to the host, if an error condition is detected by the IPPA hardware.
6	The 'IDE' bit (Interrupt on Done Enable). when set, causes the IPPA to generate an interrupt when the 'DONE' bit is set. The only exception to this is if the host issues a RESET command, which has the effect of resetting the GO, IEE, and IDE bits and setting the DONE bit.
7	The 'DONE' bit informs the host computer that the IPPA has acknowledged a previously issued command and that the IPPA is ready to accept a command. Normally, commands should not be issued unless the 'DONE' bit is set.
8	The 'RUN' bit indicates that the IPPA is in the data collection mode. When reset, the IPPA is in the HALT state.
9	The 'MICRO ERROR' bit indicates that the IPPA microprocessor has detected an error condition (i.e. software-detected error).

Table 3-3 CSR Bit Assignments (page 1 of 2)

10 The 'TIMEOUT ERROR' bit indicates that a microprocessor WAIT condition has exceeded its maximum length.

11 The 'MEMORY ERROR' bit indicates that a memory error has occurred. If this bit is set in conjunction with the "SOFT ERROR' bit, it indicates a parity error in the associative RAM. When 'MEMORY ERROR' is set in conjunction with the 'HARD ERROR' bit, it indicates either a parity error in the program RAM, a ROM access error, or an attempt to access non-existent memory.

12 The 'FIFO ERROR' bit indicates than an overflow or underflow condition exists when the host accesses the FIFO Data Buffer Register. If this bit is set in conjunction with the 'MICRO ERROR' bit, it indicates that the microporcessor has detected an operational error in the FIFO.

13 The 'PROGRAM ERROR' bit indicates either an attempt by the host to write to the CSR while the 'GO' bit is set or an attempt by the microprocessor to access the associative RAM while in the RUN mode.

14 The 'SOFT ERROR' bit indicates than an error has occurred which is deemed non-critical to IPPA operation, but which could invalidate some phase of the data collection procedure.

15 The 'HARD ERROR' bit indicates that a critical error has occurred within the IPPA hardware.

Table 3-3 CSR Bit Assignments (page 2 of 2)

CMD3 CMD2 CMD1 CMD0	Command Decode
0000	IDLE (microprocessor restart)
0001	READ TRANSFER ADDRESS
0010	READ TRANSFER COUNT
0011	READ PROGRAM MEMORY
0100	READ ASSOCIATIVE MEMORY
0101	READ PAM
0110	unassigned
0111	HALT
1000	RESET
1001	WRITE TRANSFER ADDRESS
1010	WRITE TRANSFER COUNT
1011	WRITE PROGRAM MEMORY
1100	WRITE ASSOCIATIVE MEMORY
1101	WRITE PAM
1110	unassigned
1111	RUN

Table 3-4 Command Field Summary

Access to the CSR by the Microcomputer (MC) is slightly different than the access by the host system. When the host writes the 'GO' bit, a vectored interrupt to the MC is generated. When the MC responds to this interrupt, it reads the CSR; this causes the 'GO' bit to be reset. The bits corresponding to the Command Field (bits 1-4) are the only bits read by the MC. Those bits which can be written by the MC are the 'DONE' bit, the 'RUN' bit, and the 'MICRO ERROR' bit.

3.1.2.1.2 Mask Register (MR)

The MR is a 16-bit register which is used primarily to enable or disable the counting or timing of specific events under direct control of the host system. The intent is to qualify these specific events to certain higher-level events which are detected and controlled at the system level. The MR correlates directly with the output of the associative memory, masking it on a bit-by-bit basis.

3.1.2.1.3 FIFO

The FIFO is configured with Port 1 as a low-byte and high-byte Z-BUS device and with Port 2 as a non-Z-BUS CPU device. In this configuration the MC must initialize the appropriate FIFO registers for the desired operation. It is intended tha the MC will control all data transfers through the FIFO by the appropriate interpretation of the CSR Command Field. There are several different methods which can be used to handle this exchange of data, including the following:

- o Microcomputer control using CSR
- o Host control
- o Handshake
- o Use of FIFO Message Register

The DBR provides access to a 128-word-deep bidirectional FIFO Data Buffer (DBR) and the FIFO Control and Status Register (FCS).

The DBR provides to a 128-word-deep bidirectional FIFO which allows blocks of data to be transferred between the host the the IPPA. The host controls the transfer across the Unibus interface, but the actual control of the FIFO is done by the microprocessor through the CSR command field.

3.1.2.2 Error Register (ERR)

The ERR is a 16-bit register which is accessible only to the MC. It offers a more definitive assemblage of error conditions than those available to the host via the CSR. Error conditions are defined as either hard or soft, depending on their criticality to overall MC operation. Hard errors will generate a non-maskable interrupt (NMI) which forces the MC to respond to the error condition. Hard errors can generally be regarded as fatal to the MC operation. Soft errors, on the other hand, may be recoverable or may only effect the integrity of collected data. Soft errors will generate a nonvectored interrupt (NVI), which can be enabled or disabled under software control.

Table 3-5 describes the bits in the ERR, corresponding bits in the CSR, their severity to the MC, and the cause of the error condition.

BIT	NAME	CSR BIT	HARD/SOFT	CAUSE
0	CRAM PAR ERR	11 and 14	S	Combinational Ram parity error
1	DRAM PAR ERR	11 and 14	S	Data RAM parity error
2	ARAM PAR ERR	11 and 14	S	Address RAM parity error
3	HI PAR ERR	11 and 15	H	Program RAM (hi byte) parity error
4	LO PAR ERR	11 and 15	H	Program RMA (lo byte) parity error
5	NX I/O	11 and 15	H	Attempt to access non-existent I/O
6	NX RAM	11 and 15	H	Attempt to access non-existent RAM
7	NX ROM	11 and 15	H	Attempt to access non-existent ROM

Table 3-5 Error Register Bit Descriptions (page 1 of 2)

BIT	NAME	CSR BIT	HARD/SOFT	CAUSE
8	ROM ACC ERR	11 and 15	H	Attempt to write into ROM address space
9	RAM ACC ERR	13 and 14	S	Attempt to access Associative Memory while in 'RUN' mode
10	VEC OVFL		H	Vector silo register overflow
11	FIFO ERR	12	S	Host access to FIFO causes either overflow or underflow
12	TIMEOUT	10	H	MC 'WAIT' timer has elapsed while accessing FIFO
13	OVERRUN		S	PAM did not respond in time to valid Unibus Cycle
14				undefined
15				undefined

Table 3-5 Error Register Bit Descriptions (page 2 of 2)

3.1.2.3 System Timing Controller (STC)

The STC plays a vital role in the overall operation of the IPPA. All events which are timed or counted at a relatively high frequency are done so by using an STC. The IPPA uses eight STC circuits, each containing five 16-bit counter/timers. which makes forty counter/timers available. Each counter/timer is individually configured via the STC's internal control registers. This configuration must be done by the operational software. Table 3-6 describes the function of each STC counter/timer.

STC	Timer/Counter	Function
0	CT00	Masked Associative Memory Bit 00
0	CT01	" " " " 01
0	CT02	" " " " 02
0	CT03	" " " " 03
0	CT04	" " " " 04
1	CT05	" " " " 05
1	CT06	" " " " 06
1	CT07	" " " " 07
1	CT08	" " " " 08
1	CT09	" " " " 09
2	CT10	Kernel Mode
2	CT11	SuperMode
2	CT12	Time BBSY
2	CT13	Time CPU ot Busy
2	CT14	Count MSYN
		Count MSYN for Write Cycles
3	CT15	Priority 4
3	CT16	" 5
3	CT17	" 6
3	CT18	" 7

Table 3-6 STC Counter/Timer Functions (page 1 of 2)

STC	Timer/Counter	Function
3	CT19	CPU Wait Time for Floating Point Processor
4	CT20	NPR/NPG
4	CT21	BR7/BG7
4	CT22	BR6/BR6
4	CT23	BR5/BR5
4	CT24	BR4/BR4
5	CT25	Real Time Clock 1
5	CT26	Real Time Clock 2
5	CT27	BR6/BG6
5	CT28	BR5/BG5
5	CT29	BR4/BG4
6	CT30	Instruction Fetch
6	CT31	CPU/Memory Overlap (11/45)
6	CT32	CPU/Memory Overlap (11/70)
6	CT33	Cache MISS
6	CT34	CPU Memory Cycle
7	CT35	Memory Read Cycle
7	CT36	Total Memory Cycle
7	CT37	CPU Memory Cycle/CPU Contention
7	CT38	Unibus Memory Cycle/Unibus Contention
7	CT39	MBC Memory Cycle/MBC Contention

Table 3-6 STC Counter/Timer Functions (page 2 of 2)

3.1.2.4 Interrupt Controller (IC)

The IC provides a method of generating prioritized, vectored interrupts to the MC. The IPPA uses five IC circuits, each capable of handling eight interrupt inputs. These five ICs are daisy-chained to provide the MC with forty different prioritized interrupts. The IC is made extremely versatile through the use of programmable registers, which are internal to each IC. Common or individual vectors can be assigned to each interrupt, and this information, as well as other operational parameters, is sent to each IC under control of the operational software. Table 3-7 lists the interrupt inputs to each IC, and the respective priority and group priority.

Priority Level	Devices	Function
39	CT 26	Real Time Clock
38	Host INTR	Interrupt Occurred on Unibus
37	GO H	Interrupt from Host to MC
36	PAM INTR	Interrupt from PAM to MC
35	CT 00	CRAM
34	CT 01	CRAM
33	CT 02	CRAM
32	CT 03	CRAM
31	CT 04	DRAM
30	CT 05	DRAM
29	CT 06	ARAM
28	CT 07	ARAM
27	CT 08	ARAM
26	CT 09	ARAM
25	CT 35	Memory Read Cycle
24	CT 36	Total Memory Cycles
23	CT 37	CPU Memory Cycle/CPU Contention
22	CT 38	Unibus Memory Cycle/Unibus Contention
21	CT 39	MBC Memory Cycle/MBC Contention
20	CT 30	Instruction Fetch

Table 3-7 Interrupt Table (page 1 of 2)

19	CT 31	CPU/Memory Overlap (11/45)
18	CT 32	CPU/Memory Overlap (11/70)
17	CT 33	Cache MISS
16	CT 34	CPU Memory Cycle
15	CT 10	Kernel Mode
14	CT 11	Super Mode
13	CT 12	Time BBSY
12	CT 13	Count Total Unibus Reads' and Writes'
11	CT 14	Count Unibus Write Cycles
10	CT 15	Priority Level 4
9	CT 16	Priority Level 5
8	CT 17	Priority Level 6
7	CT 18	Priority Level 7
6	CT 19	CPU Wait due to FPP
5	CT 20	NPR/NPG
4	CT 21	BR7/BG7
3	CT 22	BR6/BG6
2	CT 23	BR5/BG5 (BR4/BG5) taken at same time
1	CT 27	BR6/BG6
0	CT 28	BR5/BG5 (BR4/BG4) taken at same time

Table 3-7 Interrupt Table (page 2 of 2)

3.1.2.5 Vector Register (VECR)

The VECR is a 16-word-deep, 9-bit-wide FIFO silo which captures the lower nine bits of Unibus data whenever an interrupt occure in the host system. Whenever the VECR has been loaded with a vector, an interrupt will be generated to the MC. The VECR can store up to sixteen vectors before an overflow error condition exists. The MC operational software will normally unload the VECR in its interrupt service routine by simply reading the VECR.

3.1.2.6 Peripheral Activity Module (PAM)

The PAM is used to analyze information placed on the Unibus by the host system. It does this through a combination of information loaded into its internal registers by the MC and program routines which are contained in its internal Micro-program ROM. The PAM responds to all addresses of the I/O Page within the range of 0600-06FF (HEX). This allows direct mapping of Microcomputer addresses to the 32 internal registers of the PAM.

3.1.2.7 Serial Communications Controller (SCC)

The Microcomputer I/O address decoding provides for an optional SCC. The designated SCC is the Z8030, which is compatible with the MC and provides two full-duplex serial channels. The SCC and a small amount of support circuitry, such as RS-232 drivers and receivers, would allow the MC to communicate directly to a terminal and/or serial printer.

3.1.3 Associative Memory (AM)

The AM consists of a 64K-by-16 dynamic RAM which is partitioned into three groups: (1) a 64K-by-8 Address Associative RAM (ARAM), (2) a 64K-by-4 Data Associative RAM (DRAM), and (3) a 64K-by-4 Combinational Associative RAM (CRAM). Parity is generated and checked for each group. The AM associates in the following manner.

When the IPPA is in the RUN mode, i.e., collecting data, each of the three AM groups is addressed uniquely. The ARAM is addressed by either the sixteen bit virtual address or the upper sixteen bits of the physical address generated by the host processor. If a particular address or group of addresses is of interest, a bit in the ARAM corresponding to the address(es) of interest is preset during the user set-up session. Then, if this particular virtual or physical address is accessed by the host, the ARAM output will indicate than an association has been made. The eight-bit-wide ARAM allows eight unique associations for each address.

The DRAM is addressed by the host processor's 16-bit data bus. In a manner similar to the ARAM, the four-bit-wide DRAM allows four unique associations to be made for every data word generated by the host.

The CRAM is addressed by ten selected host processor status and control signals plus four outputs from the ARAM and two outputs from the DRAM. The four-bit-wide CRAM allows four unique associations to be made on the 65,536 different combinations of signals which address the memory.

When the IPPA is in the HALT mode, i.e., not collecting data, the MC can access the AM to either load the memory or to verify its contents. Paragraph 3.1.1.2 shows that the AM occupies Page 4 and Page 5 of the MC Address Space. The lower 32K of the AM is in the address range from 40000H to 4FFFFH and the upper 32K is in the range from 50000H to 5FFFFH. The MC may access the AM only on a word basis, and while in the HALT mode the MC generates all refresh cycles to the dynamic RAM.

While in the RUN mode the Associative Memory Timing and Control logic controls all AM operation, including refresh. The memory cycle time is 300 nanoseconds for both an association cycle or a refresh cycle. Arbitration logic determines which cycle occurs.

3.1.4 Qualifier Logic

The Qualifier Logic is responsible for reducing - or qualifying - discrete signals from other areas of the IPPA hardware and from the host system to a form that represents a unique input to the measurement collection logic, i.e., the System Timing Controller. The Qualifier Logic can be roughly separated into two sections. The first section is used with either the PDP-11/45 or 11/70, and the second section is used only with the PDP-11/70.

3.1.5 Peripheral Activity Module

The need for a Peripheral Activity Module (PAM) originally arose with the requirement to assess peripheral performance on the Unibus. Since DMA-type devices leave no telltale traces while performing NPR

transfers, the PAM was conceived to allow determination of Unibus usage by a specific device through association of the address that is placed on the Unibus by the device during an NPR transfer. The PAM calculates the address range "on the fly" by interpreting the data loaded into the peripheral device's registers prior to initiation of the DMA transfer. The algorithm used to determine the address range is as follows:

- Low Address = Contents of Bus Address Register Merged with Extension Bits
- High Address = Low Address Plus 1's Complement Word Count

At this time it is felt that the concept of analyzing Unibus information "on the fly" can be broadened to include the analysis of Unibus data transferred between the host system and any register or group of registers for a specific peripheral device. This might allow the PAM to collect a wide assortment of statistics dealing with peripheral performance and utilization.

3.2 Packaging

One of the initial - though not critical - design goals was to place all the IPPA hardware on a single DEC hex-height module, compatible with those used in the PDP-11/45 and 11/70 Central Processing Units. This was to allow the IPPA to be placed in one of the host CPU's Small Peripheral Controller (SPC) slots with a minimum impact on existing host system hardware. The hardware implementation shown in the schematic drawings is that of a somewhat expanded design, which includes embellishments for hardware and software debugging, and has not been optimized for reduction of parts count. However, even though the number of integrated circuits can easily be reduced by ten percent, it is still more realistic to assume that at least two printed circuit boards (PCBs) will be required.

The following figures, based on approximations, are provided as an estimate for the packaging requirements.

Number of components	235
Number of normalized components *	300
PCB surface area @ 0.6 IN /component	180 IN
PCB surface area @ 0.65 IN /component	195 IN
PCB surface area @ 0.7 IN /component	210 IN
PCB surface area @ 0.75 IN /component	225 IN
DEC hex-height PCB (16 1/2 IN x 7.75 IN)	127 IN

* A normalized component is designated as a 20-pin DIP integrated circuit and has a surfae area of 1 in. x 0.3 in.

The best IPPA performance will be achieved when it is physically the first device on the Unibus. For all intent this physical location is

the first SPC slot in the CPU, and this will be the recommended location.

3.3 EMI/RFI Emissions

The IPPA hardware is intended to be mounted within the physical confines of the host system and should not generate any EMI/RFI emissions which are substantially different from those of the host system. Any communication between the IPPA and an external I/O device will be done via low-speed RS-232 serial interfaces. Internal, discrete probe points will consist of twisted-pair or shielded wires similar to existing backplane wiring.

3.4 Power Requirements

The following table summarizes the power requirements for all IPPA hardware, except passive components. The circuitry for the Peripheral Activity Module is also excluded from this table. Where possible, the typical values are listed for an ambient temperature of 25°C. Any exceptions to this rule are noted. Power consumption will be dependent on the memory configuration and memory utilization. The dynamic RAM memories and the EPROMs are placed in the "standby" mode anytime the memories are not accessed, and this reduces their typical supply current by a factor of ten. The following values reflect the "active" supply currents and, thus, should represent a worst case condition.

Configuration	Supply Current	Total Power
1. Operational system (2K x 16 ROM and 16K x 18 RAM)	8.48 Amperes	42.4 Watts
2-a. Development system (2K x 16 EPROM and 16K x 18 RAM)	8.02 Amperes	40.1 Watts
2-b. Development system (16K x 16 EPROM and 64K x 18 RAM)	8.98 Amperes	44.9 Watts

Item	Part No.	Supply Current(mA)	Quantity	Total Supply Current(mA)	Total Power(mW)	Notes
1	SN74LS00	1.6	3	4.8	24.0	
2	SN74LS02	2.2	3	6.6	33.0	
3	SN74LS04	2.4	4	9.6	48.0	
4	SN74LS08	3.4	9	30.6	153.0	
5	SN74S08	25.0	1	25.0	125.0	
6	SN74LS10	1.2	2	2.4	12.0	
7	SN74LS11	3.4	1	3.4	17.0	
8	SN74LS14	10.32	2	20.64	103.2	
9	SN74LS20	0.8	1	0.8	4.0	
10	SN74LS21	3.4	2	6.8	34.0	
11	SN74LS27	2.7	2	5.4	27.0	
12	SN7428	22.52	1	22.52	112.6	
13	SN74LS30	0.48	2	0.96	4.8	
14	SN74LS32	4.0	5	20.0	100.0	
15	SN74LS51	1.1	2	2.2	11.0	
16	SN74LS73A	4.0	2	8.0	40.0	
17	SN74LS74A	4.0	20	80.0	400.0	
18	SN74S74	30.0	4	120.0	600.0	
19	SN74S86	50.0	2	100.0	500.0	
20	SN74S112	30.0	1	30.0	150.0	

Table 3-8 Power Requirements (page 1 of 3)

Item	Part No.	Supply Current(mA)	Quantity	Total Supply Current(mA)	Total Power(mW)	Notes
21	SN74LS114A	4.0	2	8.0	40.0	
22	SN74S114	30.0	2	60.0	300.0	
23	SN74LS123	12.0	1	12.0	60.0	
24	SN74LS132	7.04	2	14.08	70.4	
25	SN74LS133	0.48	1	0.48	2.4	
26	SN74LS138	6.3	4	25.2	126.0	
27	SN74S138	49.0	2	98.0	490.0	
28	SN74LS139	6.8	1	6.8	34.0	
29	SN74S153	45.0	12	540.0	2700.0	
30	SN74LS157	9.7	4	38.8	194.0	
31	SN74S157	50.0	4	200.0	1000.0	
32	SN74S158	50.0	1	50.0	250.0	
33	SN74LS161A	19.0	3	57.0	285.0	
34	SN74S174	90.0	2	180.0	900.0	
35	SN74S175	60.0	1	60.0	300.0	
36	SN74LS195A	14.0	4	56.0	280.0	
37	SN74S225	80.0	2	160.0	800.0	
38	SN74LS244	20.0	7	140.0	700.0	
39	SN74S244	110.0	1	110.0	550.0	
40	SN74LS245	55.0	4	220.0	1100.0	

Table 3-8 Power Requirements (page 2 of 3)

Item	Part No.	Supply Current(mA)	Quantity	Total Supply Current(mA)	Total Power(mW)	Notes
41	SN74LS260	4.4	2	8.8	44.0	
42	SN74S280	67.0	5	335.0	1675.0	
43	SN74LS373	24.0	3	72.0	360.0	
44	SN74LS374	27.0	8	216.0	1080.0	
45	SN74LS378	13.0	2	26.0	130.0	
46	DS8641	50.0	14	700.0	3500.0	
47	Z8001A	300.0	1	300.0	1500.0	MAX
48	AmZ8127	125.0	1	125.0	625.0	
49	Z8038A	250.0	2	500.0	2500.0	MAX
50	Am9513	160.0	8	1280.0	6400.0	
51	Am9519-1	80.0	5	400.0	2000.0	
52	Am25LS2518	17.0	5	85.0	425.0	
53	Am25LS2521	27.0	2	54.0	270.0	
54	HM-7616	180.0	2	360.0	1800.0	MAX
*54	2716	57.0	2-8	114.0-456.0	570.0-2280.0	
*54	2732	85.0	2-8	170.0-680.0	850.0-3400.0	
55	NMC5295	35.0	18	630.0	3150.0	
**55	NMC4164	45.0	18	810.0	4050.0	MAX
**55	2118	23	18	414.0	2070.0	
56	NMC4164-1	45.0	19	855.0	4275.0	MAX
57	CO-238A					

* indicates optional parts

** indicates alternate parts

Table 3-8 Power Requirements (page 3 of 3)

3.5 Qualifier Logic (Sheets (S) and (T))

The Qualifier Logic on Sheet (S) is used with both PDP-11/45 and 11/70 host systems. U217-U219A, B (SN74LS08) are used to qualify the outputs of the Associative Memory (AM 000-03 H, AM006-07 H, and AM D12-15 H) with the Mask Register, MASK00-09 H. These qualified outputs, MAM 00-09, go directly to the System Timing Controller logic. U220 (74LS73A) is used in conjunction with the lower four bits of the Associative Memory to provide special gating to one of the STC circuits, U118, on Sheet (K). The GATE 1 H and GATE 3 H signals are asserted when MAM 01 H and MAM 03 H, respectively, are generated by the Associative Memory and Mask Register. The gate signals are negated when MAM 00 H and MAM 02 H are generated in a similar manner. This special gating provides the programmable option of timing the interval between any two events which can be generated by the Combinational Associative Memory. U223A (SN74LS74A) and associated logic gates are used to assess the interaction between the host processor and the host system's Floating Point Processor (FPP). When executing a floating point instruction, the FPP acts as a co-processor and can operate almost entirely independent of the host processor. When execution of a floating point instruction begins, the host processor causes START FP H to be generated. This in turn allows U223 A to be clocked into its active state, asserting FP REQ L, which enables U1368 (SN74LS02). If the FPP does not respond within the arbitrary time of one microsecond, DLY REQ L is generated which asserts FP WAIT H. This signal is used as a gating signal in the System Timing Controller (Sheet (K)) to measure the time that the CPU waits for the FPP, an indirect measure of the FPP efficiency. During normal execution of a floating point instruction, the FPP issues FP SYNC L in response to the host processor's FP START(I) H or FP ATTN L. FP SYNC L is inverted and enables U222A (SN74LS260) to reset U223A. Other conditions which reset U223A are: (1) HALT H, (2) FP EXC TRAP L, which indicates that the FPP has generated a "trap" to the host

processor, and (3) INTR H, which will normally abort the execution of a floating point instruction to allow the host processor to service an interrupt.

The host processor's priority and current mode are decoded by U221 (SN74LS139) to produce PRIOR 4-7 L and USER L, SUPER L, and KERNEL L, U219C (SN74LS08) is used to generate UB OVERLAP H, an indication of overlapping operation between the host processor and other Unibus devices.

The logic on Sheet (T) is used to assess the operation the PDP-11/70 memory system. U225-U227 (SN74LS04 and SN74LS14) buffer various signals from the host system which are used in the IPPA hardware. U90C (SN74LS10) asserts START H when any memory operation is initiated. At the next negative transition of 24M CLKH, U229A (SN74LS73A) is clocked into its active state, asserting MEM CIP H which indicates that a memory cycle is in progress. MEM CIP H is used to clock flip flops U2248, U228A and U228B. When a memory cycle starts, one of these flip flops will have an asserted signal at its "D" input - CP ACKN M, MBC ACKN H, OR UB ACKN H - and one flip flop will consequently be clocked into its active state to indicate the current memory user. The signals generated by these flip flops enable U230 and U231 (SN74LS51) to assess contention for memory resources. For example, if either a Unibus or MBC memory cycle is in progress, the assertion of CP REQ H will generate CP CNTH L, indicating that the CPU is contending for memory. U231 is also used to determine when CPU operation and other memory access overlap. The signal 70 OVERLAP H is asserted when this condition exists.

U232A (SN74LS21) is used to determine a cache memory "miss" by ANDing the necessary conditions - a slow, read memory cycle not initiated by an MBC.

U233 (SN74LS195A) is used to generate the clock for the CPU Data Latch (Sheet (O)) and the CPU Status Latch (Sheet (R)). When the CPU accesses memory, CP REQ L is asserted which places U233 in the "parallel load"mode. On the next positive transition of 20M CLK H a "one" will be loaded into either Q_A or Q_B, depending on READ H. During a write cycle READ H will be negated, causing U1360 (SN74LS02) to load Q which immediately generates CLK CPD M. During a read cycle READ H will be asserted, causing U2008 (SN74LS08) to load Q_A. When the memory cycle is completed, data is strobed into the CPU and CP REQ L is negated. This places U233 in the "shift" mode, and approximately 200 nanoseconds later a "one" is shifted into Q , generating CLK CPD H. This allows time for the data to become valid at the input to the CPU Data Latch.

3.6 Peripheral Activity Module (Sheet (U))

The Peripheral Activity Module (PAM) is essentially a co-processor in the 1PPA hardware, whose sole purpose is to collect information regarding peripheral devices through interpretation of address and data information that is presented to the Unibus. The hardware used to implement this function is shown on Sheet (U). Although this schematic is meant to serve as an intermediate hardware design, it is functionally accurate.

The PAM utilizes an extremely fast 16-bit microprocessor integrated circuit, the Am29116. In this application the Am29116 performs its intended function by executing a series of short programs which are contained in the Micro-program ROM. The Micro-program ROM is controlled by the 2910 Micro Sequencer. The particular program being executed depends on several factors: (1) information loaded into the PAM's internal registers by the Microcomputer, (2) current "status" of the PAM, (3) current inputs to the PAM's Jump ROM and Condition Code Multiplexer, and (4) current Unibus Latch data.

When power is applied to the IPPA hardware, the PAM is initialized by a power-up routine contained within the Micro-program ROM. This serves the same purpose as the Microcomputer's power-up routine. It places the hardware in a known state ready to respond to external inputs. After the PAM is initialized, it will enter an "idle" state, where it awaits a command from the Microprocessor. Whenever the Microprocessor accessed the PAM, ENB PAM L is asserted which generates PAM REQ H. When the RAM is in an idle state, it uses this input to the Condition Code Multipexer to branch to one of its micro-programs, the micro-program is determined by the inputs to the Jump ROM, which are derived from the Microbus (Sheet (G)). If the PAM cannot respond to the Microcomputer within the time allocated for a normal memory access, BUSY L will be asserted causing PAM WAIT to be generated. This will cause WAIT cycles to be generated by the Microcomputer until the PAM can properly respond.

The information presented to the Jump ROM can be thought of as a type of "macro" instruction to the PAM. Tehse macro instructions are divided into two broad categories: load class instructions and execute class instructions. A load class instruction is used to pass data between the Microcomputer and the PAM registers. When the Microcomputer performs a write cycle, PAM REQ H is asserted and READ DATA L is negated. READ DATA L controls the direction of the Microbus Data Transceivers (SN74LS245), which when enabled by the Micro-program ROM (ROM 19 L) allow data to be transferred from the Microbus (MDB 00-15 H) to the Pambus (PAMBUS 00-15 H). When a load class macro instruction occurs and the Microcomputer is performing a read cycle, the PAM will place the proper data on the Pambus, which is transferred to the Microbus by the combination of signals ROM 19 L and READ DATA L.

An execute class macro instruction will place the PAM in its data collection mode; this corresponds to the IPPA hardware being placed in the RUN mode. When the RUN mode is entered, the logic on Sheet (D) is enabled, and UB CYCLE L will be asserted for every valid Unibus cycle. This will generate PAM REQ H to the Condition Code Multiplexer. Since the PAM has been placed in the data collection mode, PAM REQ H will cause the PAM to branch from an idle state to the appropriate micro-program routine. In this routine Micro-program ROM bits 20 and 21 control the transfer of information from the Unibus Address Latch (UBAL 01-16 M) and the Unibus Data Latch (UBDL 00-18 H) to the Pambus where this information can be processed by the PAM. When the micro-program determines that an interrupt should be generated to the Microcomputer - for example, when an internal counter overflows - the Micro-program ROM will assert PAM INTR H, and an interrupt will be generated by the Interrupt Controller (Sheet (M)). The appropriate Microcomputer Interrupt Service Routine will generate the required load class macro instruction to read PAM information, as described before.

4.0 THEORY OF OPERATION

4.1 Overview

This section describes the detailed Theory of Operation for the IPPA hardware. It is intended to be used as a narrative to the schematic drawings. The hardware can be roughly modularized into the following groups:

- o Unibus Interface
- o Microcomputer
- o Associative Memory
- o Qualifier Logic
- o Peripheral Activity Module

Each module is discussed fully in the following paragraphs.

4.1.1 Schematic Drawing Notation

An attempt has been made to increase the readability and comprehensibility of the schematic drawings through the use of mixed logic, polarized mnemonics, assertion level descriptors, and logic state indicators.

4.1.1.1 Mixed logic refers to the drawing of gate symbols to reflect their intended logical function, rather that the function designated by the integrated circuit manufacturer. Thus, a gate designated as a

NAND gate by the manufacturer, such as the SN7400, may appear in the schematic drawings as either a NAND gate or an OR gate (with inverted inputs), depending on the intended logical function. Figure 4-1 shows some equivalent mixed logic gates.

4.1.1.2 Signal mnemonics suggest the logic function performed by the signal at its active assertion level. All signal mnemonics carry an assertion level descriptor, either L or H, as the final character. As the name implies, this descriptor indicates the level of the signal when it is asserted, either LOW of HIGH. An overbar is associated only with a mnemonic, not with the assertion level descriptor. The overbar indicates that the complement of the logic function suggested by the mnemonic is performed at the active assertion level indicated by the accompanying descriptor.

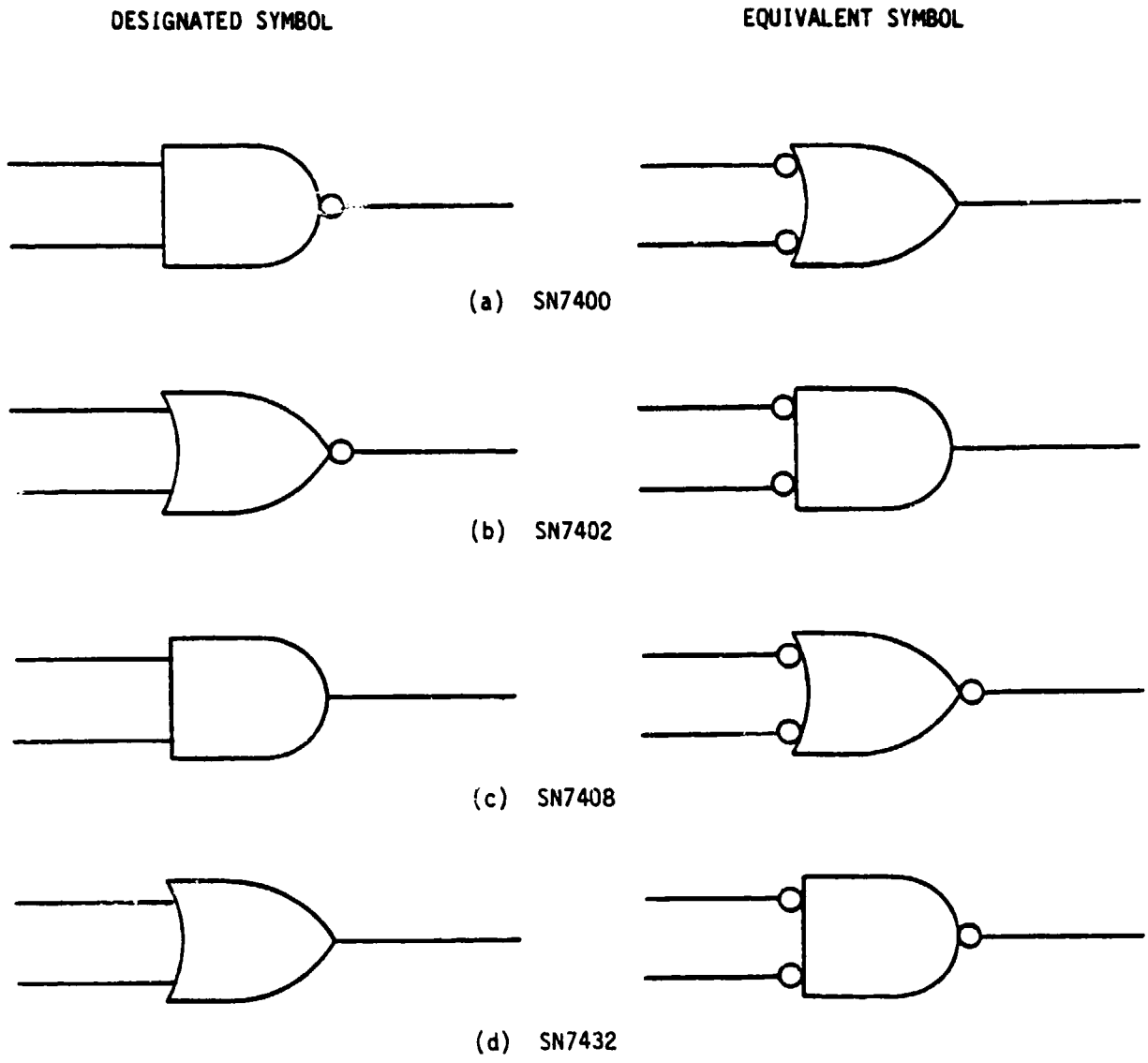


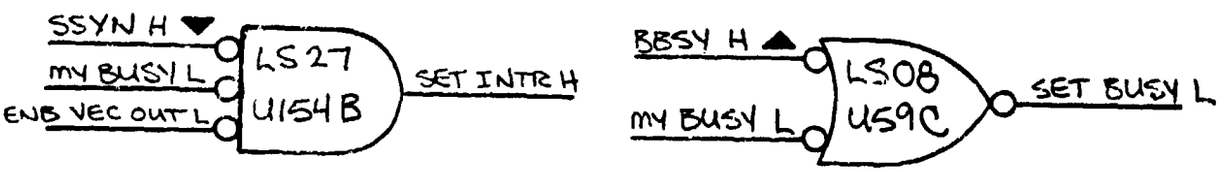
Figure 4-1 Mixed Logic Gates

4.1.1.3 Logic state indicators (or logic level indicators) are the "bubbles" at inputs or outputs of logic elements; they convey information concerning the asserted polarity of the signal on the connecting logic line. The presence of the indicator shows that the assertion level is LOW, and the absence of the indicator shows that the assertion level is HIGH. Since logic state indicators and assertion level descriptors do not always match, some guidelines were established by which the descriptors and indicators have been assigned in the schematic drawings.

- At the OUTPUT of a logic element, the state indicator and assertion level descriptor ALWAYS indicate the same level.

- At the INPUT to a logic element, the state indicator and assertion level descriptor will normally indicate the same level. However, if an incompatibility does exist, a small triangle (▼,▲) is used at the input to indicate that this incompatibility is intentional. The triangle serves two purposes: first, it is an immediate indication to anyone interpreting the schematic that the incompatibility is intentional, and second, it gives insight into the funtion of the signal.

4.1.1.4 For logic gates, the triangle indicates that the complement of the signal is to appear in the output logic expression. When the ▼ symbol appears at the input of an OR gate, it enables the logic function performed by the gate. The following example refers to gates found on Sheet (F) of the schematic drawings.



The AND function of gate U154 B in disabled when SSYN H is asserted. The output logic expression for this gate is:

$$\text{SET BUSY L} = (\overline{\text{SSYN}} \cdot \text{MY BUSY} \cdot \text{ENB VEC OUT})\ \text{H}$$

Note that the SSYN input mnemonic has been complemented in the output expression. The OR function of gate U59C is enabled when BBSY H is asserted. The output logic expression for this gate is:

$$\text{SET BUSY L} = (\overline{\text{BBSY}} + \text{MY BUSY})\ \text{L}$$

Again note that the BBSY input mnemonic is complemented in the output expression.

4.1.1.5 For other logic elements the ▼ symbol indicates that the function of the associated input is inhibited by the presence of the signal and, conversely, enabled by its complement. For example, on Sheet (A) of the schematic drawings, flip flop U14B has the signal MSYN H ▼ on its "CLR" input. This implies that the "clear" function is inhibited by MSYN H and enabled by its complement, MSYN L.

4.1.1.6 Flip flops, especially D-type, always seem to add an element of obscurity when attempting to

interpret schematic drawings. The guidelines for determining the operational characteristics of flip flop from the schematic notation is straight-forward and follows the same guidelines described in the paragraphs above.

- Internal mnemonics are those assigned by the manufacturer and are used to describe the internal mechanism of the flip flop according to the manufacturer's data sheet.

- Signal mnemonics, assertion level descriptors, and state indicators describe the functional operation of the flip flop.

- Signal mnemonics suggest the function of the ACTIVE STATE of the flip flop, in almost all cases.

- The active state of the flip flop is determined by the absence or presence of a state indicator at the clocked input of the flip flop. The absence of the indicator shows that the "Q" output will be HIGH when the flip flop is in its ACTIVE state. (The flip flop is generally regarded as "set" in this condition.) The presence of a state indicator at the input shows that the "Q" output will be LOW when the flip flop is in its ACTIVE state. (The flip flop is generally regarded as "reset" in this condition.) The functions of the PRESET and CLEAR inputs follow the definition of the active state.

- State indicators on the "Q" and "Q" outputs always coincide with the assertion level descriptor of the

associated signal mnemonics when the flip flop is in its ACTIVE state.

- The ▼ symbol at a clock input indicates that the flip flop will be clocked at the complementary transition of the signal connected to the clock input.

- The ▼ symbol at a data input indicates that the associated signal will inhibit the function of that input.

4.2 Unibus Interface (Sheets (A) thru (F))

The Unibus Interface consists of the following logic circuitry:

- Unibus Address Interface and Decode Logic
- Unibus Data Interface, Internal Data Bus, Control Register
- Unibus Control Interface
- Mask Register, Unibus Latch
- FIFO, Vector Register
- Unibus Acquisition Logic

4.2.1 Unibus Address Interface and Decode Logic (Sheet (A))

The Unibus address lines, BUS A00-17 L, are received and buffered by bus transceivers U1-U5 (DS8641) for use in the IPPA hardware. The block of addresses used to communicate between the host system and the

IPPA is determined by the setting of switch SW1, a 10-pole DIP switch. When the host communicates with the IPPA, UBA13-17 H will be asserted, indicating a Unibus Peripheral Page address, and UBA03-12 H will correspond to the selected address SWA03-12 H. U6 and U7 and cascaded 8-bit comparators (Am25LS2521) which generate the signal MY ADRS L when the IPPA is addressed. This signal is used with UBA02 H and C1 H to generate the individual read and write signals to the appropriate registers. When the FIFO is accessed, FIFO H is asserted and U14 (SN74LS74A) generates either WR FIFO L or RD FIFO L, providing that the proper FIFO access recovery time has elapsed. FIFO recovery is discussed in the following paragraph.

4.2.2 Unibus Data Interface, Internal Data Bus, Control Register (Sheet (B))

The Unibus data lines, BUS D00-15 L, are received by bus transceivers U15-U18 (DS8641) to transfer data information to the IPPA hardware. Incoming data, UBD00-15 H, is routed to the various registers and to the Unibus Data Latch. The bus transceivers also transmit data from the Internal Data Bus, IBD00-15 H, to the Unibus. The tri-state Internal Data Bus is the source of all data going to the Unibus, and it can be enabled with data from the following sources:

- Control and Status Register (Sheet (B) and (D))
- Mask Register (Sheet (D))
- FIFO Control Register (Sheet (E))
- FIFO Data Buffer (Sheet (E))
- Vector (Sheet (F))

Data from the Internal Data Bus is transmitted to the Unibus when either of two conditions exist: (1) the IPPA Unibus Interface is generating an interrupt to the host and ENB BED OUT L is asserted, or (2) the host system is performing a read access to one of the IPPA registers, and MY SSYN L is asserted and C1 H is negated. (Remember that the darkened triangle associated with a single name indicates that the mnemonic should be complemented in the output or terminal logical expression).

The lower byte of the Control and Status Register (CSR) is implemented using D-type flip flops. U19 and U21 (SN74LS74A) store CSR bits 0, 5, 6, and 7, and U20 (Am25LS2518) stores the four bits of the Command Field, CSR bits 1-4. These CSR bits, with the exception of bit 7, are written when the host system performs a write acces to the CSR, generating CLK CSR H. When bit 0, the "GO" bit, is written, DONE H is reset and an interrupt to the IPPA Microcomputer will be generated. The Microcomputer Interrupt Service Routine (MISR) will direct the Microcomputer to read the CSR, which generates MRD CSR L. This signal resets GO H and enables the tri-state outputs of U20, which source MBD01-04 H on the Microbus. (The Microbus is discussed in paragraph 3.1). The MISR will then proceed according to what is reads in the Command Field, and at some point in time it should direct the Microcomputer to write bit 7, the "DONE" bit in the CSR. DONE H is set when MWR CSR L is negated, which insures that MBD07 H is valid when the flip flop is clocked.

Any attempt by the host system to write to the CSR while Go H is asserted will set PGM ERR H, indicating a programming error.

FIFO access is controlled by shift register U22 (SN74LS195A) and flip flop U23 (SN74LS74A). This logic is required to meet certain timing specifications of the Z8038 integrated circuit. The access sequence is initiated whenever the host system performs a read or write

operation to either the FIFO Control Register or the FIFO Data Buffer. This action will generate RD FIFO L or WR FIFO L, respectively, and LD RECOVER L will load U22 with all ones. RECOVER H is delayed two clock periods (333 nanoseconds) before generating CLK SSYN H to allow FIFO data to be accessed. On Sheet (C) CLK SSYN H clock set MY SSYN L, which is delayed 50 nanoseconds before DLY SSYN H asserts BUS SSYN L on the Unibus. Th: delay insures that Unibus data is valid when BUS SSYN L is asserted. If a write operation is in progress, MY SSYN L will terminate FIFO access by clearing WR FIFO L on Sheet (A).

Returning to Sheet (B), the second part of the access sequence can now be discussed. Until FIFO access is terminated, LD RECOVER L keeps U22 loaded with ones, asserting RECOVER L. On Sheet (A) RECOVER L at the input of U13 inhibits FIFO access by disabling this gate. When a FIFO access terminates, LD RECOVER L is negated and U22 is placed in the shift mode. Zeros will be shifted in via the J and K inputs, and four clock periods later (667 nanoseconds) RECOVER L is negated, enabling FIFO access.

4.2.3 Unibus Control Interface (Sheet (C))

The Unibus Control Interface receives the remaining Unibus signals for the use within the IPPA and transmits those signals necessary for Unibus acquisition when the IPPA generates an interrupt to the host system. U1 and U28-U32 (DS8641) perform these tasks. U33A (SN74LS28) is the buffer for the IPPA hardware initialization signal, RESET L. This signal is asserted when either the host system generated BUS INIT L, causing INIT H to be asserted, or the System Clock Generator (Sheet (F)) generates MICRO RST L.

The write clock for the Vector Register (Sheet (E)) is generated by the Schmitt Trigger U27D (SN74LS132), which is configured as a 75-nanosecond delay element. Whenever a host-system interrupt is

generated, INTR H will be asserted. This will produce WR VECR H, provided that MY B SY L is not asserted, indicating that the IPPA is generating the host-system interrupt.

The prioity at which the IPPA operates is determined using a standard DEC Priority Jumper Plug. It converts REQ L, GRANT IN H, and GRANT OUT H to correnponding Unibus signals BUS BRn L, BUS BGn IN H and BUS BGn OUT H.

If the host system accesses either the CSR or the Mask Register, U24A will be enabled and U26A will be direct-set. Since there is essentially no access time for these registers, no delay is required before setting MY SSYN L.

4.2.4 Mask Register, Unibus Latch (Sheet (D))

The Mask Register, U34-U37, consists of four quad D-type registers with both TTL and tri-state outputs (Am25LS2518). The TTL outputs are used in the Qualifier Logic (Sheet (S)), and the tri-state outputs source the internal Data Bus when the host system is reading the Mask Register and RD MASK L is asserted. The Mask Register is loaded by CLK MASK H, which is generated when the host system writes data, UBD00-15 H, into the Mask Register.

U38-U41 ISN74LS374) and U26B (SN74LS74A) are latches for the Unibus data, UBD00-15 H, and address, UBA0 -1 1. The Unibus Latch, as well as U42, are clocked by CLK UBL H. This signal is asserted at a point when both Unibus data and address signals are valid. For a Unibus DATO (data out) operation, this occurs when MSYN H is asserted. For the Unibus DATI (data in) operation, this occurs 75 nanoseconds after SSYN H is asserted. U42 (SN74LS74A) generates UB CYCLE H whenever the IPPA is in the RUN mode and CLK UBL H occurs. UB CYCLE H is used to inform the Peripheral Activity Module (PAM - Sheet U) that a valid

Unibus cycle has occurred. IF the PAM does not respond with UB ACK L before the next Unibus cycle occurs, then OVERRUN H will be set, indicating an error condition.

The Control and Status Register (CSR) is read by the host system when tri-state buffers U43 and U44 (SN74LS244) are enabled by RD CSR L. This allows U43 and U44 to source the Internal Data Bus. CSR bit 11 is derived from MEM ERR H, a logical OR of all detected memory error conditions.

4.2.5 FIFO, Vector Register (Sheet (E))

The FIFO, U47 and U48, consists of two Z8038A FIFO circuits and additional support logic. Certain aspects of the Z8038 which relate to its external operational interface will be discussed here. The reference material in Appendix A should be consulted for a more thorough description of the hardware and software operation of the Z8038. The mode pins, M0 and M1, configure the Port 1-side of U47 as a Z-Bus High Byte device and the Port 1-side of U48 as a A-Bus Low Byte device. This, in essence configures the FIFO for 16-bit word transfers and allows the IPPA Microcomputer to access the FIFO as a peripheral device. It is critical to note that the Z8038 must be configured under software control so that Port 2 responds as a Non-Z-bus device and that both Port 1 and Port 2 transfer data by means of "synchronized CPU and FIO". This will allow the support logic to function in its intended manner.

The operation of the Port 2-side is straight-forward. When the host system accesses the FIFO, FIFO L will be asserted along with either RD FIFO L or WR FIFO L. UBA01 H will determine which register - FIFO Data Buffer Register or FIFO Control and Status Register - is accessed. During a read operation the tri-state data lines of the Z8038 will be enabled and will source the Internal Data Bus. During a

write operation WR FIFO L enables tri-state buffers U45 and U46, which allow Unibus data UBD00-15 H to be written into the FIFO. If the host system attempts to write into a full or read from an empty Data Buffer Register, XFER ERR L will be asserted. This, in turn, sets FIFO ERR H, indicating an error condition.

The operation of the Port 1-side of the FIFO is controlled by the Microcomputer. Since the Z8038 is configured as Z-Bus device, its data lines, MAD00-15 H, are connected directly to the unbuffered, multiplexed address/data lines of the Z8001A Microprocessor. The other control signals come from the Microbus, which is described in Paragraph 3.1. The major difference in accessing the FIFO from the Port 1-side is that the Microcomputer can directly address all sixteen registers internal to the Z8038, whereas the host system can directly address only two, as described above.

At this time, it is intended that all FIFO operations be controlled by the Microcomputer via comands written into the SCR by the host computer. The Microcomputer accesses the FIFO when it addresses any FIFO register, thereby generating SEL FIFO L.

The registers internal to U47 (high byte) are addressed by MBA01-04 H, and the registers internal to U48 (low byte) are addressed by MAD01-04 H. Identical information will be presented on these address lines when they are strobed by the Z8038, so the same register in each device is always selected. Read and write operations are determined by MBWR L and are always performed on a 16-bit-word basis by the Microcomputer. Since the registers in each Z8038 are 8-bits-wide, those registers which control the internal operation must be loaded with the same eight bits of information; i.e., the high byte and the low byte of the data word must be identical. Specifically the registers which must be accessed in this psuedo byte manner are as follows:

REGISTER	MICROBUS ADDRESS (HEX)	ACCESS TYPE
Control Register 0	70200	READ/WRITE
Control Register 1	70202	READ/WRITE
Control Register 2	70213	READ/WRITE
Control Register 3	70214	READ/WRITE
Interrupt Status Register 0	70204	READ/WRITE
Interrupt Status Register 1	70206	READ/WRITE
Interrupt Status Register 2	70208	READ/WRITE
Interrupt Status Register 3	7020A	READ/WRITE
Byte Count Register	7020E	READ ONLY
Byte Count Compare Register	70210	READ/WRITE

The registers which use only the low byte of data are as follows:

REGISTER	MICROBUS ADDRESS (HEX)	ACCESS TYPE
Interrupt Vector Register	7020C	READ/WRITE
Pattern Match Register	7021A	READ/WRITE
Pattern Mask Register	7021C	READ/WRITE

The remaining registers are used to transfer 16-bit words between the two ports of the Z8038. They are as follows:

REGISTER	MICROBUS ADDRESS (HEX)	ACCESS TYPE
Message Out Register	70216	READ/WRITE
Message In Register	70218	READ/WRITE
Data Buffer Register	7021E	READ/WRITE

To add a degree of versatility to the Port 1-side of the FIFO, two features have been implemented. First, the Z8038 interrupt mechanism operates at the same level as the Microcomputer Interrupt Controller circuitry (refer to Paragraph 3.6). This allows the FIFO to generate vectored interrupts for a variety of reasons, when enabled. Second, the internally-generated WAIT signal is used in a slightly different manner. If the Microcomputer attempts to write into a full or read from an empty Data Buffer Register, BUFFER WAIT L will be asserted by the Z8038. This signal is used to activate a Wait Timer. Since the Microcomputer is ultimately a slave to the host system, the Wait Timer gives more flexibility to host system - Microcomputer interaction. BUFFER WAIT L triggers U53B (SN74LS123), the Wait Timer, and both TIMEOUT L and TIMEOUT H are negated. This enables U55A (SN74KS32) to generate FIFO WAIT L, which causes extra WAIT cycles to be inserted in the current Microcomputer memory cycle. If BUFFER WAIT L is still asserted at the expiration of the Wait Timer's timing interval, U54A (SN74LS74A) will be clocked into its active state, asserting TO ERR H and indicating an error condition. The assertion of TIMEOUT H by the Wait Timer disables FIFO WAIT L, allowing the Microcomputer to complete its current memory cycle. Since U53B is retriggerable, it is only at the end of its timing interval that the state of BUFFER WAIT L becomes relevant. In what can be regarded as normal operation, the FIFO WAIT L signal will be controlled exclusively by BUFFER WAIT L, and U54A will continue to be clocked into its inactive state by TIMEOUT H.

The Vector Register, U49 and U50, consists of two 16-by-5 FIFO memories (SN74S225), operating in parallel. When a host system interrupt occurs, the vector is placed on the Unibus data lines UBD00-08 H. This data is clocked into U49 and U50 by WR VECR H. When data is present, LO RDY H and HI RDY H will be asserted, generating HOST INTR H which interrupts the Microcomputer at the highest Vectored Interupt level. When the Microcomputer reads the Vector Register, RD VEcR L is asserted. This enables the tri-state outputs of U49 and U59 which now source Microbus data lines MBD00-08 H. If more than one vector has been written into the Vector Register, the absolute level of HOST INTR H will follow the absolute level of RD VECR L. Thus, when the Microcomputer completes its read operation, RD VECR L will be negated and HOST INTR H will again be asserted. This transition of HOST INTR H is mandatory for the proper operation of the Interrupt Controller circuitry. If the Vector Register is full when WR VECR H is asserted, U52A (SN74S74A) will be clocked into its active state and VECR OVFL H will be asserted, indicating an error condition.

4.2.6 Unibus Acquisition Logic (Sheet (F))

The Unibus Acquisition logic allows the IPPA to generate interrupts to the host system. Operation begins with all flip flops on Sheet (F) reset; all but U58B, which is self-resetting, are reset by RESET L. The host system must enable interrupts by setting bit 5 and/or bit 6 in the CSR. These CSR bits appear at the "D" inputs to U56 (SN74LS74A) and enable the Unibus Acquisition logic. When the corresponding clock signal occurs, i.e., either CLK ERR H or DONE H, U55B (SN74LS32) will assert GET BUS H. When U57A (SN74LS114A) is clocked, MY BR H is set and, on Sheet (C), the appropriate BUS BRn L will be asserted. When the host Priority Arbitration logic issues a grant at the corresponding level, and when this grant is passed to the IPPA as BUS BGn IN H, it will be asserted as GRANT IN L at the input of inverter U11E. The inverted signal, GRANT H, disables U55C

(SN74LS32) and clocks U58A (SN74LS74A). With MY BR H asserted at its "D" input, U58A is set in its active state, asserting MY SACK H and MY SACK L. MY SACK H will assert BUS SACK L (Sheet (C)) and also resets MY BR H at the next clock to U57A. Whenever U57A is reset $\overline{\text{MY BR H}}$ is asserted, indicating that the IPPA is not requesting the Unibus. $\overline{\text{MY BR H}}$ at the "D" input to U58B will generate GRANT OUT L whenever the flip flop is clocked by GRANT H. U58B will be reset when the incoming grant, GRANT IN L, is negated. In this manner U58B acts to "pass the grant" whenever the IPPA is not requesting the Unibus.

Returning to the case where the IPPA has generated the bus request, MY SACK L is asserted, which clears U56 and also enables U60A (SN74LS32). If or when BBSY H is negated, indicating that the Unibus is not in use, SET BUSY L will be asserted. This direct-sets both U57B and U201A (SN74LS114A); U57B asserts MY BUSY H, which generates BUS BBSY L (Sheet (C)), and MY BUSY L. The tri-state outputs of buffer U61 (SN74LS244) are enabled, and the Internal Data Bus is sourced with the vector selected by SW2, a 7-pole DIP switch. MY BUSY L also maintains the assertion of SET BUSY L and enables one input of U154B. ENB VEC OUT L enables the low-byte data transceivers on Sheet B , placing the vector on Unibus data lines BUS D00-07 L, enables a second input to U154B, and also disables the clocked resetting of U201B. If or when SSYN H is negated by the previous Unibus operation, U154B will be enabled, and U201B will be clocked into its active state, generating MY INTR H and MY INTR L. The assertion of MY INTR H causes BUS INTR L (Sheet (C)) to be placed on the Unibus and also enables one input to U59D. MY INTR L disables the clocked resetting of U57B and also enables U55C. When the incoming grant is negated, GRANT H will also be negated. This allows U55C and U59B to generate CLR SACK L, which resets U58A and negates SET BUSY L.

When the host system has read in the vector, it will issue BUS SSYN L, which causes SSYN H to be asserted and enables U59D. On the next

clock to U201A, the flip flop will be reset, negating ENB VEC OUT L and removing BUS INTR L from the Unibus. Finally, on the following clock, U57B will be reset, removing BUS BBSY L from the Unibus. This completes the interrupt cycle.

4.3 Microcomputer (Sheets (G) thru (M))

The Microcomputer consists of the following logic circuitry:

- o Microprocessor and System Clock Generator
- o Program ROM
- o Program RAM
- o I/O Control and Error Register Buffer
- o System Timing Controller
- o Interrupt Controller

4.3.1 Microprocessor and System Clock Generator (Sheet (G))

The Z8001A is the 6-Megahertz, segmented version of the Z8000 Microprocessor. Those aspects of the Z8001A which are relevant to the IPPA hardware will be discussed in this paragraph. A more thorough description of the Z8000 can be obtained by consulting those reference documents listed in Section 6.

The use of the circuitry associated with the Microprocessor is similar to that developed for standard applications. MAD00-15 H are the multiplexed address/data lines from the Z8001. U66-U68 (SN74S373) are octal, tri-state latches which use the buffered Address Strobe, BAS H, as the "enable" input. When BAS H is negated, the Microbus Address, MBA0018 H, is latched for the current Microcomputer cycle. The Microbus is defined as the collection of buffered address, data, and control signals which are used to transfer information within the Microcomputer. The segment outputs SEGMT0-2 H are used as the upper three addresses, MBA16-18 H, of the Microbus. The outputs of U66-U68,

as well as the remaining Microbus buffers, are enabled by ENB BUS L. Since no devices currently in the IPPA hardware are capable of requesting use of the Microbus, ENB BUS L will normally be asserted. U69 and U70 are octal, bidirectional bus transceivers (SN74LS245) which buffer data between the Microprocessor, MAD00-15 H, and the Microbus, MBD00-15 H. When DATA STB L is asserted by the Microprocesor, it indicates that the multiplexed address/data lines are being used to transfer data. The use of this signal in the timing of data transfers will be discussed in Paragraph 3.3. The direction of data transfer is controlled by WRITE L; when asserted, it indicates a transfer from the Microprocessor, to the Microbus. When WRITE L is negated and BDS H is asserted, READ DATA L is asserted; this directs the data from the Microbus to the Z8001 during a read operation. U65 is an octal buffer (SN74S244) for other Microbus control signals.

The Z8001 status lines, STATUS 0-3 H are decoded by U63 (SN74LS138) and U62, the System Clock Generator (AmZ8127), to determine the type of transaction currently in progress in the Z8001. When STATUS 3 H is asserted, U63 is disabled and the Microcomputer is accessing its memory space for a data transfer. When STATUS 3 H is negated, U63 decodes the transaction - either a dynamic refresh cycle or an interrupt acknowledge cycle. STATUS 3 H is buffered by U65 and U225A to provide MEM CYC H and MEM CYC L for use elsewhere in the Microcomputer. The System Clock Generator, U62, generates all clock signals used in the Microcomputer. It generates ZCLK H, the 6-Megahertz clock used only by the Z8001, and it provides general-purpose system clocks at the following frequencies: 24.0 Megahertz, 6.0 MHZ, 3.0 Megahertz, and 1.5 Megahertz. U62 also controls system initialization by asserting MICRO RST L, either during power-up or in response to MAN RESET L; this signal is asserted by pushing SW3, the manual reset switch.

If certain devices in the Microcomputer cannot respond of perform properly in the period normally allocated, MICRO WAIT L will be

asserted. This causes U62 to assert WAIT L to the Microprocessor, which will then insert WAIT cycles in the current transaction until WAIT L is negated. U62 has an internal counter which generates an internal timeout signal if MICRO WAIT L is asserted for more than 16 clock cycles (4.0 microseconds). This signal will generate a Non-Maskable Interrupt only if TIMEOUT L is asserted.

This allows the FIFO Wait Timer to extend the timeout interval to its period. A Non-Maskable Interrupt will also be generated when HARD ERR H is asserted (Sheet (J)), indicating that a critical hardware error has been detected. The Microprocessor's Non-Vectored Interrupt input is controlled by SOFT ERR L, which when asserted indicates that a non-critical error has been detected. The Vectored Interrupt input is the remaining input which is used on the Z8001. An interrupt request, VEC INTR L, can be made by either the FIFO (Sheet (E)) or the Interrupt Controller (Sheet (M)).

4.3.2 Program ROM (Sheet (H))

The Microcomputer address space is decoded into 64K-byte "pages" by U72 (SN74S138). When MEM CYC H is asserted (ENB BUS L is normally asserted), the Microcomputer is accessing its memory space and MBA16-18H are decoded to generate an enable signal to the corresponding page. U89C (SN74LS00) asserts MC REQ H if the Microcomputer accesses the Associative Memory.

Before discussing the program ROM circuitry, a brief mention of the possible PROM options should be made. The logic has been implemented to accept both 2K x 8 and 4K x 8 pin-compatible PROMs. The logic also allows the substitution of MOS EPROMs for standard bipolar PROMs. The error detection logic for Non-Existent ROM is configurable for the type and number of PROMs installed.

These options are configured using one patch plug, PP1, which is connected as follows:

- For 2K x 8 PROMs, MBA12-13 H are connected to ROM ENBO-1 H, respectively, and +5VDC is connected to ROM A12 H.

- For 4K x 8 PROMS, MBA12-14 H are connected to ROM A12 H and ROM END-1 H, respectively.

- For a 2K x 16 address space, connect MBA12-14 H to ERR SELO-12 H, respectively. For 4K x 16 address space, connect MBA13-14 H to ERR SEL1-2 H, respectively, and connect ERR SELO H to GROUND. For an 8K x 16 address space, connect MBA14 H to ERR SEL2 H and connect both ERR SELO-1 H to GROUND. For a 16K x 16 address space, connect ERR SELO-2 to GROUND.

- For PROMs with an access time greater than 220 nanoseconds, add the number of WAIT cycles required for valid access by removing GROUND from WAIT 1-3 H, as appropriate.

  WAIT 1 H = 385 nanoseconds maximum access time

  WAIT 1-2 H = 550 nanoseconds maximum access time

  WAIT 1-3 H = 715 nanoseconds maximum access time

The Page 0 enable signal from U72, ENB ROM L, enables the Program ROM circuitry during a memory access. ENB ROM L enables U8B (SN74LS139), which decodes ROM ENB 0-1 H in order to select one-of-four pairs of ROM integrated circuits, and it also enables one iput of both U136B (SN74LS02) and U10C (SN74LS02). If an attempt is made to write into

Page 0, U136B will assert BAD WR H and U74A (SN74LS74A) will be clocked set when the address strobe, MBAS L, is negated. This asserts ROM ACC ERR H, indicating an error condition.

When MBMREQ L is asserted, two actions are initiated. First, U75D (SN74LS32) is enabled, which will generate the enable signal, ENB ROMn L, corresponding to the select signal, SEL ROMn L, asserted by U83. The combination of these two signals will enable the tri-state outputs of one pair of PROMs, which will provide the data word to the Microbus, MBD00-15 H, from the location addressed by MBA01-11 H and ROM A12 H.

Second, when MEM CYC L is asserted, U10C generates ROM ACCESS H. If an address is accessed outside the address space created by the ROM Plug, PP1, U10D will assert CLK NX ROM H, which clocks U54B (SN74LS74A). This asserts NX ROM H, indicating as error condition. ROM ACCESS H also places U76 (SN74LS195A) in the "shift" mode. If the ROM Patch Plug is configured to generate WAIT cycles, U76 will have been loaded with the number of ones corresponding to the number of desired WAIT cycles. At the next transition of the clock input, 6M CLK H, a one will be shifted into the "Q " output, and its complementray output "Q " will assert ROM WAIT L. U203A (SN74LS21) responds by asserting MICRO WAIT L to the System Clock Generator (Sheet (G)), until a zero is shifted into the "Q " output and ROM WAIT L is negated. This allows the Microprocessor to complete it memeory cycle, at whcih time MBMREQ L will be negated. When ROM ACCESS H is negated, U76 will once again be loaded with the value at its parallel inputs.

4.3.3 Program RAM (Sheet (I))

Page 2 and Page 3 in the memory address space have been allocated for Program RAM. Access to either page will either ENB LO PM L or ENB HI PM L. U51D (SN74LS00) then asserts ENB PM H, which enables the

Program RAM circuitry. The RAM itself consists of eighteen 16K x 1 or 64K x 1 dynamic RAM integrated circuits' U91-U108. These are separated into two equal groups - the low byte and the high byte - each containing eight data bits and one parity bit. the Program RAM can be acessed on a byte or word basis by the Microprocessor. Byte and word access is controlled by U89A and U89B (SN74LS00). When MBWORD L is asserted by the Microprocessor, indicating a word access, both ENB LO BYTE H and ENB HI BYTE H are asserted, enabling both the low and high byte of Program RAM. When MBWORD L is negated, only one byte will be enabled, and this is determined by the state of MBA00 H. The Z8000 accesses the high byte at even addresses.

The address to the RAM circuits, PRAM A0-7 H, is multiplexed by U86 and U87 (SN74S157). When 16K x 1 RAM circuits are used, PRAM A7 H is disregarded. The "A" inputs to U86 and U87, MBA01-08 H, are used as the row address to the RAM. The "B" inputs, which are used as the column address, are selected by the RAM Patch Plug, PP2. When 16K x 1 RAM circuits are used, MBA08-14 H will be selected as the "B" inputs, and 16K MODE L will be connected to GROUND. When 64K x 1 RAM circuits are used, MBA09-15 H will be selected as the "B" inputs, and 16K MODE L will be connected to +5VDC.

When either a valid memory read/write cycle or a refresh cycle is in progress, MBMREQ L will be asserted by the Microprocessor. MBMREQ L is used as the row address strobe to the RAM. Since SEL COL H is currently negated, U86 and U87 select the row address as PRAM A0-7 H, and this is strobed into the internal row address latch. U27C (SN74LS132) delays MBMREQ L 40 nanoseconds and then asserts SEL COL H, which causes U86 and U87 to select the column address as PRAM A0-7 H.

If a valid read/write cycle is in progress, the Microporcessor will next assert BDS H. This is the buffered Data Strobe, a timing signal the Microprocessor uses to control data transfers. Since ENB PM H is asserted, BDS H will be the remaining enabling input to U90 and U90B

(SN74LS10). This causes LO CAS L and HI CAS L to be asserted, depending on the state of ENB LO BYTE H and ENB HI BYTE H, respectively. LO CAS L and HI CAS L are the column address strobes to the RAM. The assertion of LO CAS L and/or HI CAS L cause the column address, now selected by U86 and U87, to be strobed into the internal column address latch. The "CAS" signal controls the internal mechanism of the RAM. During a write operation, indicated by the assertion of MBWR L, data from the Microbus, MBD00-15 H, is strobed into RAM when the "CAS" signal is negated. During a read operation the assertion of the "CAS" signal also enables the tri-state output buffers, so that the RAM now sources data to the Microbus.

U110 and U111 (SN74S280) continuously generate an even-parity signal for the Microbus data, MBD00-15 H. During a write operation the Microprocessor will place a word (byte) on the Microbus and U110 and U111 will assert or negate GAN HI PAR H and GEN LO PAR H accordingly. These signals are connected to the "Data In" inputs of RAM circuits U99 and U108. When the "CAS" signal is negated, these parity bits will be strobed into the RAM, along with the RAM, along with the Microbus data.

During a read operation U99 and U108 generate RD HI PAR H and RD LO PAR H at the same time that the remaining RAM circuits source the Microbus; U110 and U111 again generate parity for the Microbus data. U112A and U112B (SN74S86) perform an exclusive-OR between the parity bit which is read from U99 and U108 and the parity signal which is currently generated by U110 and U111. If the parity is dissimilar, U112 will assert HI PAR BAD H and LO PAR BAD H accordingly.

These signals are inputs to U91A and U91B (SN74LS08), and if they are asserted during a read operation, U88 (SN74LS74A) will be clocked into an active state at the end of the memory cycle by the negation of HI CAS L and LO CAS L, indicating an error condition. U88 is self-latching, so the error indication will remain until it is cleared

by the Microcomputer. MBWR L at the input of U91 inhibits error detection during write operations.

U109A and U109B (SN74LS02) are used to detect non-existent addresses when 16K x 1 RAM circuits are used. While in the 16K mode any address access beyond 23FFF(HEX), the 32K-byte boundary, will cause U74B (SN74LS74A) to be clocked into its active state, indicating an error. U74B is clocked whenever Program RAM is accessed and when the Microbus Address Strobe, MBAS L, is negated.

4.3.4 I/O Control and Error Register Buffer (Sheet (J))

The I/O Control logic decodes the address space for the Microcomputer Peripheral Page and generates various control signals. Primary address decoding is done by U77B (SN74S260) which asserts VALID I/O H when MBA11-15 H are all negated. VALID I/O H enables U113 (SN74S138) and disables U73B (SN74LS27), which prevents clocking U117A (SN74LS74A). This flip flop, when set, indicates an access to non-existent I/O. When U113 is enabled by ENB I/O L and MEM CYC L, the Microcomputer is addressing one of its peripheral devices; U113 decodes MBA08-10 H and asserts an enable signal to one-of-eight devices.

U114 (SN74LS138) is enabled by ENB IC L to decode MBA05-07 in order to select one of the five Interrupt Controller integrated circuits (Sheet (M)).

Because of certain timing specifications for both the FIFO (Sheet (E)) and the System Timing Controllers (Sheets (K) and (L)), access to these devices is performed in a controlled manner. Shift registers U116 and U204 (SN74LS195A) function as timers in teh access sequence to insure that valid access and recovery times are met. U116 controls recovery for both the FIFO and the STC, while U204 controls access time which is only required for the FIFO. Both U116 and U204 are initially cleared by RESET L, so RIP H, ENB WAIT L, and I/O WAIT L are negated. If either ENB FIFO L or ENB STC L is asserted, U91C (SN74LS08) will assert SLOW L. This allows U60B (SN74LS32) to assert LD TIMER L, which parallel-loads U116. Since a zero is loaded in from the "D" input, both ENB WAIT L and RIP H remain negated. If ENB FIFO

L is asserted,U60C will assert SEL FIFO L and the Microprocessor will access the FIFO. ENB FIFO L will also allow U136C (SN74LS02) to assert FIFO ACC H, which puts U204 in the "shift" mode. The next two clocks to U204 will shift ones into "Q_D", and its complementary output will assert I/O WAIT L. This generates two WAIT cycles in the memory cycle.

If ENB STC L is asserted to initiate the access sequence, U115 (SN74LS138) is immediately enabled to decode MAB05-07 H in order to select one of the eight System Timing Controller circuits. The Microprocessor performs this access during a standard memory cycle.

When the access is completed, SLOW L will be negated. This puts U116 in the "shift" mode, and on the next clock a one will be shifted into "Q_D", asserting both ENB WAIT L an RIP H. RIP H disables U60B, U60C, and U136C; ENB WAIT L disables one "enable" input to U115 while it enables one input to U109C (SN74LS02). This state is maintained until U116 shifts a zero from its J-K input into "Q_D". If another access is made and SLOW L is asserted, U109 C will assert WAIT H. Regardless of the device being accessed, RIP H will disable U136 C, which prevents the possible assertion of FIFO ACC H and keeps U204 in the "load" mode. At the next clock to U204, a one will be parallel loaded into "Q_D" and I/O WAIT L will subsequently be asserted, causing the Microporcessor to insert WAIT cycles in its standard memory cycle. This will continue until the timing interval of U116 expires (1 microsecond). If this second access is to the FIFO, U204 will immediately be placed in the "shift" mode when RIP H is negated, and the memory cycle will be extended for an additional two WAIT cycles, as before, to allow the FIFO to be accessed.

U205 (SN74S133) and U206 (SN74LS30) generate the clocks to U207. When any "hard" error occurs, CLK HD ERR H will be asserted, clocking U207A into its active state. HARD ERR L generates a Non-Maskable Interrupt to the Microprocessor. U207A is cleared when the Microporcessor

status decoder (Sheet (G)) asserts NMI ACK L. When any "soft" error occurs, U206 will assert CLK SFT ERR H, which clocks U207B into its active state. SOFT ERR L generates a Non-Vectored Interrupt to the Microporcessor. U207B is cleared when the Microporcessor status decoder asserts NVI ACK L.

U208 and U209 (SN74LS224) make up the Error Register Buffer. When the Microporcessor reads the Error Register, RD ERR L is asserted; this enables the tri-state outputs of U208 and U209, which now source the Microbus MBD00-15 H. RD ERR L at the "D" input to U210 A (SN74LS74A) causes the flip flop to be clocked into its active state, asserting CLR ERR H. When access to the Error Register is completed, RD ERR L is negated, which allows U51B (SN47LS00) and U20CB (SN74LS08) to assert CLR ERR L. This signal, which is also generated by RESET L, clears the error-latching flip flops used in the IPPA hardware.

U211A and U211B (SN74LS32) generate the special signals, I/O RD L and I/O WR L, which are used primarily by the System Timing Controller and Interrupt Controller circuits.

4.3.5 System Timing Controller (Sheets (K) and (L))

As described in the previous section, the Microprocessor accesses one of the System Timing Controller circuits (Am9513), U118-U125, by asserting SEL STC0-7 L. These signals, along with I/O RD L, I/O WR L, and MBA01 H, control read and write operations to the circuits. The internal tri-state bus buffer can either accept data from or source data to the Microbus, MBD00-15 H.

The Am9513 has twenty-one internal registers which function as control and status registers and as "data" registers. These registers are used to configure the operation of the Am9513. Reference material has been included in Appendix A which provides a more thorough description of the hardware and software operation.

After each of the five internal 16-bit counters has been configured, its operation relies primarily on three external signals. Two input signals will provide a "source" and a "gate" to the counter, and an "out" signal will be generated when the counter overflows. This "out" signal generates an interrupt request to the Interrupt Controller.

4.3.6 Interrupt Controller (Sheet (M))

The Interrupt Controller, U126-U130, utilizes five Am9519-1 integrated circuits. The Am9519 has eight internal registers plus a vector memory, which are used to control its operation. These registers are accessible to the Microcomputer when it addresses the Interrupt Controller and asserts SEL IC0-4 L. Data is transferred via the lower byte of the Microbus, MBD00-07 H; I/O RD L, I/O WR L, and MBA01 H control the transfer. Appendix A contains reference material pertaining to hardware and software operation of the Am9519.

Internally, IREQ0 has the highest priority; externally, the five circuits are cascaded so that U126 has the highest priority, followed by U127, etc. When an external interrupt request is made - or multiple interrupts - VEC INTR L will be asserted by the Interrupt Controller. When the Microprocessor status decoder asserts VI ACK L, the Interrupt Controller asserts IC WAIT L and WAIT cycles are inserted in the Microcomputer interrupt cycle, if necessary. The internal mechanism of the Am9519 allows the vector corresponding to the highest priority unmasked pending request to be asserted on the Microbus. When the vector is asserted, IC WAIT L will be negated, allowing the Microcomputer to complete the interrupt cycle. The Interrupt Controller uses RESPONDING L as an internal control signal.

4.4 Associative Memor (Sheets (N) thru (R))

The Associative Memory consists of the following logic circuitry:

- o Timing and Control Logic
- o Addressing Logic
- o Address Associative Memory
- o Data Associative Memory
- o Combinational Associative Memory

The Associative Memory operates in two modes. In the "HALT" mode the Associative Memory is accessed as Page 4 and Page 5 in the Microcomputer's memory address space. Control and access is nearly identical to that of the Microcomputer Program RAM (Sheet (I)). From an operational standpoint the only difference is that the Associative Memory can only be accessed on a word basis. As with the Program RAM all refresh is controlled by the Microprocessor.

In the "RUN" mode Associative Memory access is controlled completely by its own timing and control logic. The basic memory cycle has a 300-nanosecond period, but the access cycles overlap to allow some of the Address and Data Associative Memory outputs to address the Combinational Associative Memory.

Figure 4-2 shows the Simplified Block Diagram, which is common for all three memory groups. The RAM can be addressed by the Microcomputer (in the "HALT" mode), the Refresh Counter, or the source for the actual association. Data can be transferred between the RAM and the Microbus (in the "HALT" mode), or RAM data can be latched during an association cycle for use by the Qualifier Logic.

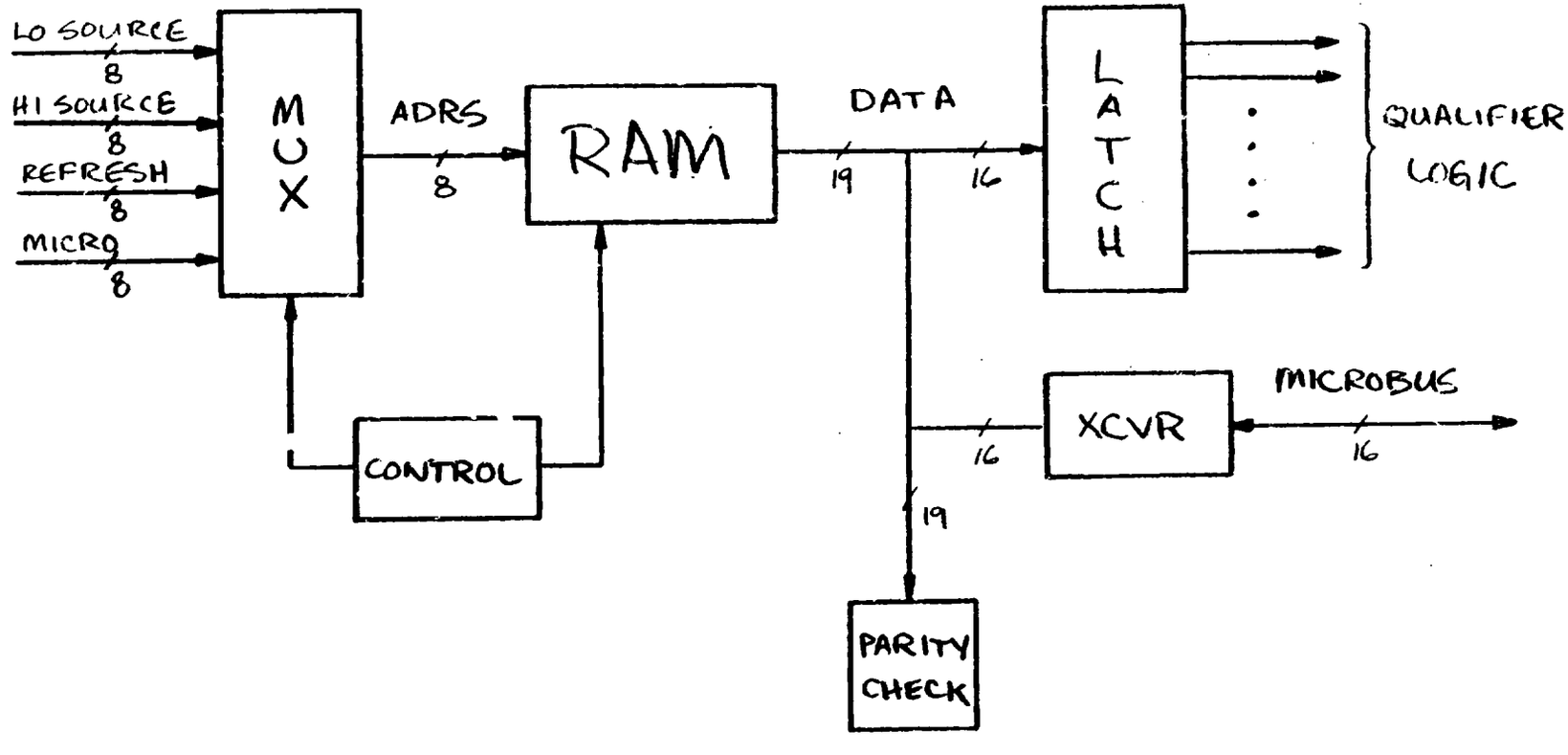


Figure 4-2 Associative Memory - Simplified Block Diagram

4.4.1 Associative Memory Timing and Control Logic (Sheet (N))

Timing within the Associative Memory is generated by flip flops U131-U134 (SN74S74, SN74S112, SN74S114). With the exception of U131A, these flip flops are initialized in their inactive state by either RESET L or HALT 2 L. U131A produces the complementary 20-Megahertz clock signals, 20M CLK L and 20M CLK L, which are derived from a 40-Megahertz oscillator. The "RUN"/"HALT" mode of the IPPA hardware is controlled by U133B and U212B (SN74LS74A). U212 stores CSR bits 8 and 9, which are written by the Microcomputer when NWR CSR L is negated. U133B synchronizes the output of U212B with the Associative Memory Timing and Control Logic.

4.4.1.1 Microcomputer Access

With U132 and U133 reset, U135C negates STATE 6 H which disables U135D; this negates the Microcomputer-access-inhibit signal from U71F, INH MC L. This keeps U13A (SN74LS74A) clocked in its active state with ENB MC H asserted. This enables U136D, asserting ENB MEM L, which enables multiplexer U139 (SN74S158). U139 generates the "RAS" and "CAS" signals to the dynamic RAM circuits. RUN H is the input-select signal to U139; in the "HALT" mode the "A" inputs are selected. These inputs are controlled by U135B (SN74LS08), U109D and U136A (SN74LS02).

Anytime the Microcomputer accesses Associative Memory, MC REQ H will be asserted. A secondary function of MC REQ H is to enable the locking of U117B (SN74LS74A). If CLK ACC H is generated while in the "RUN" mode, the assertion of RUN H causes U117B to be clocked into its active state. This asserts RAM ACC ERR H, indicating an error condition. The primary functio of MC REQ H allows the dynamic RAM to respond to the Microcomputer's current read/write cycle by enabling both U109D and U136A. The assertion of MBMREQ L will then generate MC RAS H, and the assertion of MBDS L will then generate MC CAS H. U139

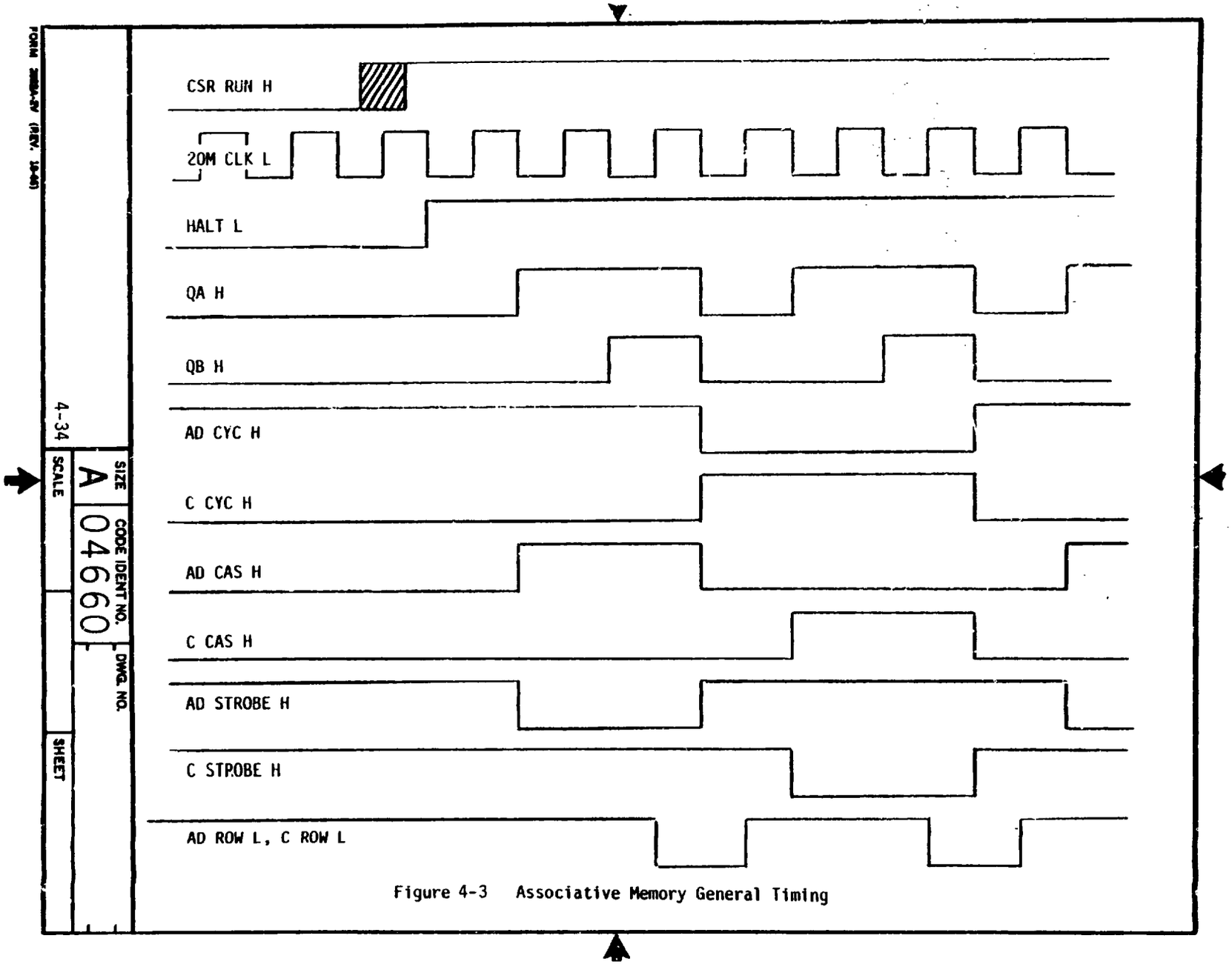


Figure 4-3 Associative Memory General Timing

uses these signals to simultaneously assert AD RAS L and C RAS L, followed by AD CAS L and C CAS L. U135B allows the assertion of RFSH L to also enable U109D. This causes MC RAS H to be generated during all Microcomputer memory refresh cycles.

4.4.1.2 Access by Timing and Control Logic

Figure 4-3 has been provided as a general-purpose timing diagram to show the relationship among various timing and control signals and may be referred to during the following discussion.

U132 and U133A make up a synchronous divide-by-six counter, which gives the Associative Memory its 300-nanosecond cycle. All critical timing is referenced to or synchronized with this counter. As was previously mentioned, the counter is initially disabled by HALT 2 L, but as soon as the "RUN" mode is entered, the counter is enabled. To clarify the discussion the six 50-nanosecond periods of this counter will be referred to as STATE 1-6, and the transitions into the states will be referred to as MT1-MT6.

Sometime after the host system writes a "RUN" command into the CSR Command Field, the Microcomputer will actually write the "RUN" bit into U212B. CSR RUN H is asserted and on the next clock to U133B, RUN H asserted and HALT H, HALT 1 L and HALT 2 L are negated . Since CSR HALT H is now negated, U135D (SN74LS08) is disabled and ENB HELT H is negated. This keeps U133B set until the Microcomputer clears the "RUN" bit. This action will be discussed shortly.

The transition from the "HALT" mode to the "RUN" mode is critical to the proper operation - and data integrity - of the dynamic RAM circuits. This transition, however, is a natural outcome of the Microcomputer's setting of the "RUN" bit in the CSR. CSR RUN H is asserted when MBMREQ L is negated, and the next possible Microcomputer memory cycle will not begin for well over 100 nanoseconds. Within 50

nanoseconds of the assertion of SCR RUN H, U133B is clocked and the mode transition takes place. Since the transition is a result of the Microcomputer addressing its I/O Page, the transition adheres to all timing and access specifications for the dynamic RAM circuits.

In the "RUN" mode the dynamic RAM is continuously accessed for either a data cycle, where an association is made, or a refresh cycle. U131B, U137B (SN74LS74A), and associated gates detemine the type of access cycle. Whenever teh host System transfers data, CLK CPD H will be asserted which clocks U137B into its active state; this asserts DATA REQ H. During STATE 6 the "D" input to U131B is sampled. If DATA REQ H is asserted, and if a refresh cycle is not required, indicated by $\overline{\text{DEMAND RFSH}}$ H, then U131B will be clocked into its active state, asserting DATA CYC H. In essence, U131B will be clocked into its active state, asserting DATA CYC H. In essence, U131B functions as an arbitrator for the next memory cycle. DATA CYC H enables counter U140 (SN74LS161A), which will negate $\overline{\text{DEMAND RFSH}}$ H in approximately 10.5 microseconds, if it is not cleared by the negation of DATA CYC H. U141A and U141B are partially enabled by DATA CYC H, which allows U134 (SN74S114) to generate "CAS" signals to the memory circuits. DATA CYC H also controls one of the "select" inputs to the memory address multiplexers, which are discussed in Paragraph 3.5. When U131B is clocked into its inactive state, a refresh cycle will be generated to the memory. Since the arbitration does not occur until STATE 6, a refresh cycle will always be the first cycle performed when the "RUN" mode is entered.

U133A and U134 generate the "RAS" and "CAS" signals to the memory. AD RAS L is asserted during STATE 1-3, and C RAS L is asserted during STATE 4-6. If a data cycle is being performed, AD CAS L will be asserted during STATE 2-3, and C CAS L will be asserted during STATE 5-6. When U134 negates a "CAS" signal, the appropriate strobe signal, AD STROBE H or C STROBE H, is generated to the Associative Memory Latch.

The transition from the "RUN" mode to the "HALT" mode is also critical to the operation of the dynamic RAM circuits. When the Microcomputer clears CSR bit 8, U212B negates CSR RUN H and asserts CSR HALT H. During STATE 6 U135D asserts ENB HALT H. This causes two actions to occur. First, U71F asserts INH MC L, which clears U137A and places it in its inactive state. This negates ENB MC H, removing one of the enabling inputs to U136D. Second, the "K" input to U13 B is now enabled, and at time MT1 the flip flop is clocked into its inactive state. This asserts both HALT H and HALT L and negates RUN H, removing the remaining enabling input to U136D. This negates ENB MEM L, which disables U139 and temporarily prevents access to the dynamic RAM circuits. Most important is that this action takes place at the end of a memory access cycle.

Now in the "HALT" mode, U135D is again disabled by the negation of STATE 6 H, and ENB HALT H and INH MC L are also negated. This allows U137A to be clocked into its active state at the end of the next Microcomputer memory cycle when MBMREQ L is negated. ENB MC H once again causes U136D to assert ENB MEM L, and access to the Associative Memory is enabled. This action insures that a minimum of 150 nanoseconds will pass between access by the Associative Memory Timing and Control logic and access by the Microcomputer.

4.4.2 Associative Memory Addressing Logic (Sheet 0)

The Addressing Logic provides the three different sources for the Associative Memory. While in the "HALT" mode, U152 and U153 (SN74LS157) multiplex the Micorbus address signals, MBA01-16 H, to provide MO AO-7 H. This is done in a manner similar to that used with the Program RAM.

Binary counters U150 and U151 (SN74LS161A) provide the refresh address, RFSH AO-7 H, when the "RUN" mode is entered; the counters are initially cleared by HALT L. In their operational mode the counters

are incremented at the end of every refresh cycle, so that all locations in the dynamic RAMs are refreshed. RFSH CYC H enables the counters, and STATE 8 H provides the clock pulse.

U146-U149 (SN74LS374) latch the address and data currently accessed by the host system's Central Processing Unit. Data from the CPU console, DISP DOO-15 H, is clocked into U148 and U149 by CLK CPD H. The latched data, CPU DOO-15 H, is then used to address the Data Associative Memory (Sheet (Q)) during an association cycle.

The CPU address, CNSL ADRSGO-15 H, is clocked into U146 and U147 by CLK CPA H. The latched address, CPU AOO-15 H, is then used to address the Address Associative Memory (Sheet (P)) during an association cycle. The source for the address latch is selected by 2-to-1 multiplexers U142-U145 (SN74S157). U73C (SN74LS27) decodes DISP ADRS SELO-2 H and asserts SEL PA H when the Address Display Select Switch on the host system console has been places in the "PROG PHY" - Program Physical - position. The "B" inputs to U142-U145 will now select the upper-sixteen physical address signals, DISP AD06-21 H. When the Address Display Select Switch is in any other position, the lower-sixteen (virtual) address signals, VAOO-03 H and DISP AD04-15 H, are selected by the "A" inputs to U142-U145.

4.4.3 Associative Memory (Sheets (P) thru (R))

The three sections of the Associative Memory - ARAM, DRAM, and CRAM - can be discussed collectively because their physical operation is the same. One Sheets (Q) and (R) U213 is used to generate a 50-nanosecond window in which the row address to the dynamic RAMs is selected and strobed into the appropriate circuits. The two flip flops are identical but are required to provide fan-out to the twelve RAM address multiplexers, U155-U158, U173-U176, and U187-U190 (SN74S153). On Sheet (Q) U33D (SN7428) provides the fan-out for the other multiplexer input-select signal, BUF RFSH H. When DATA CYC H is

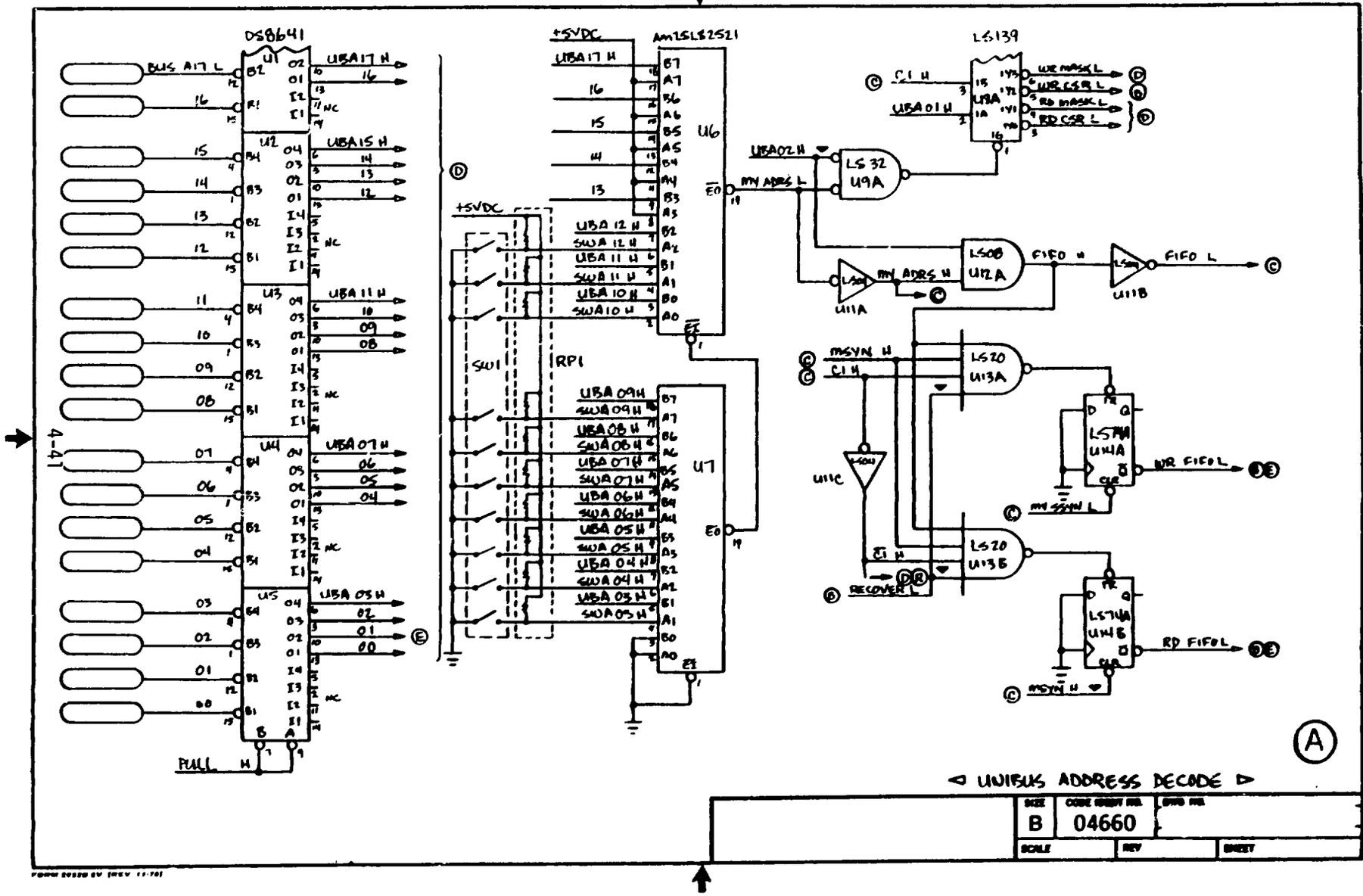
negated BUF RFSH H is asserted.

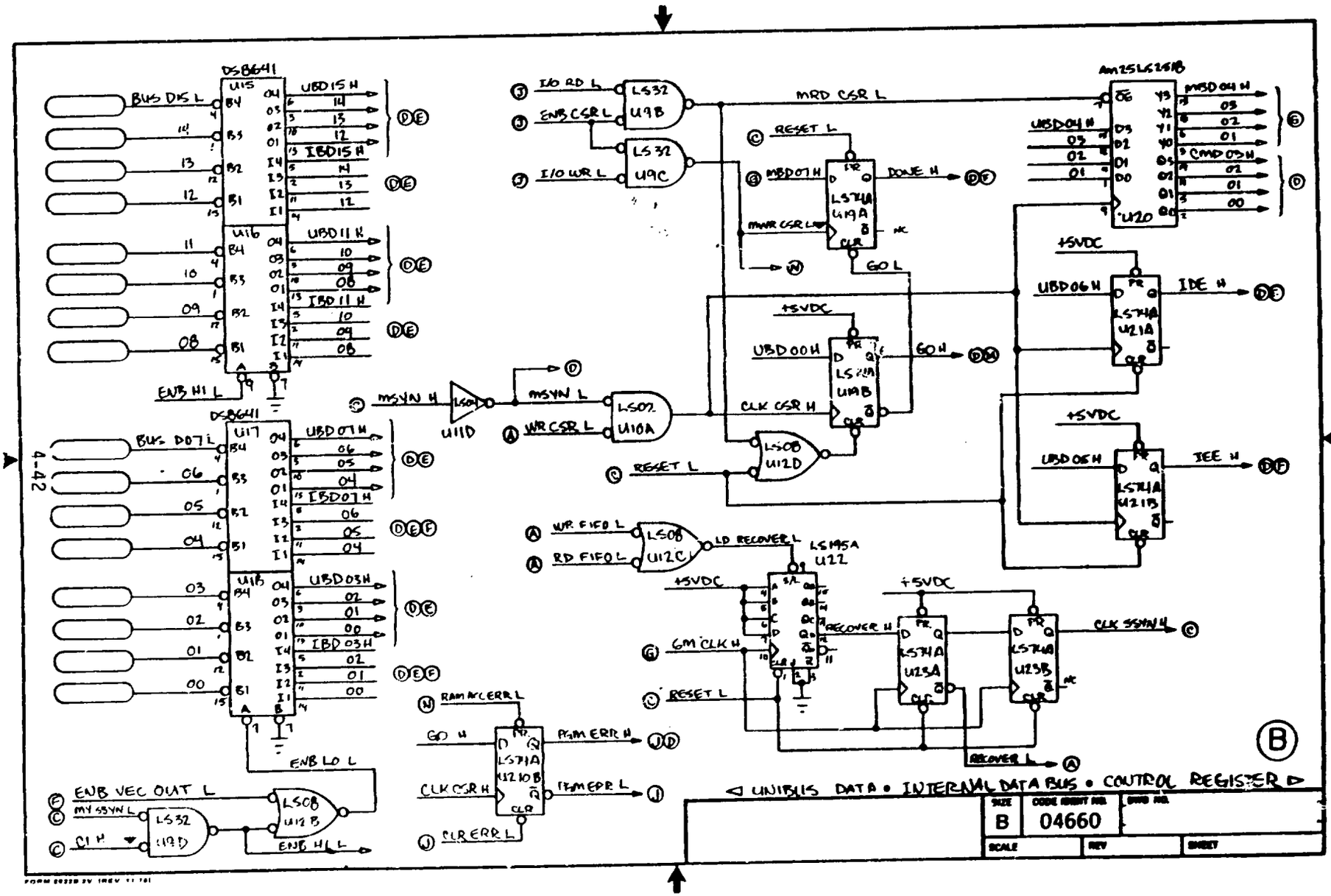
When the IPPA is in the "HALT" mode, HALT L maintains both U131B (Sheet (N)) and U213 in their inactive state. This causes RFSH CYC H to be asserted and negates both AD ROW L and C ROW L. With both input-select signals at a HIGH level, the multiplexers select their "1C3" and 2C3" inputs, MC AO-7 H, which originate at the Microcomputer. During a valid Microcomputer access to the Associative Memory, all inputs to U24C (SN74LS10) on Sheet (P) will be HIGH, and ENB MC ACC L will be asserted. This enables tri-state transceivers U168 and U182, which buffer the data between the Microbus, MBDOO-15 and the corresponding memories, CRAM DO-3 H, DRAM DO-3 H, and ARAM DO-7 H. The direction of the transfer is controlled by RD DATA L. Parity is generated and checked in a manner identical to that used with the Program RAM (Paragraph 3.3). On Sheet (R) U60D (SN74LS32) insures that WR ENB L is asserted only when in the "HALT" mode.

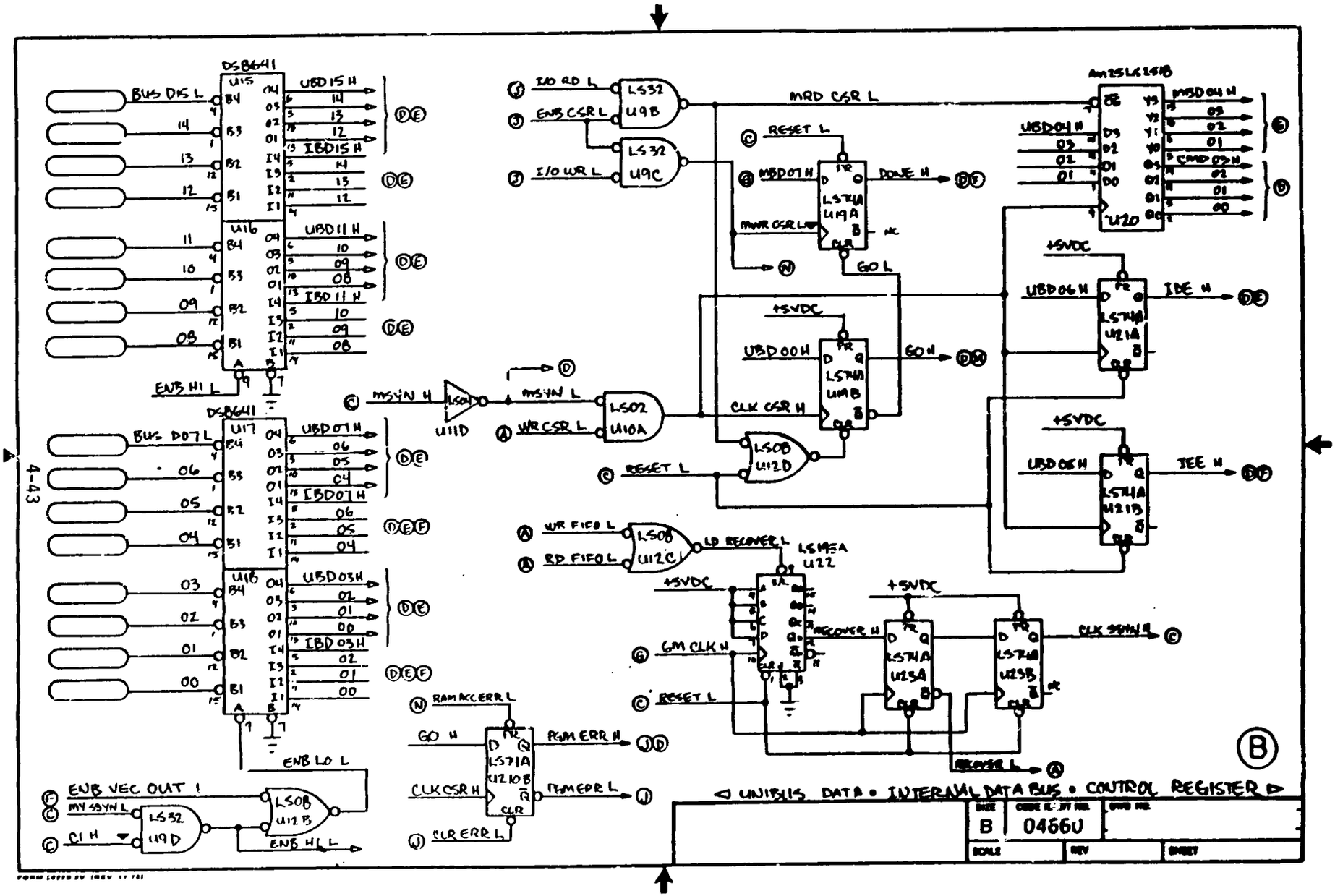
When the "RUN" mode is entered, the type of cycle is determined by the timing and control logic described in Paragraph 3.5. During a refresh cycle, BUF RFSH H is asserted, and the "RAS" signal is asserted. This causes the multiplexers to select their "1C2" and "2C2" inputs, RFSH AO-7 H, the refresh address. Since the "CAS" signals are not generated, the address presented to the dynamic RAM circuits outside the row address window is inconsequential.

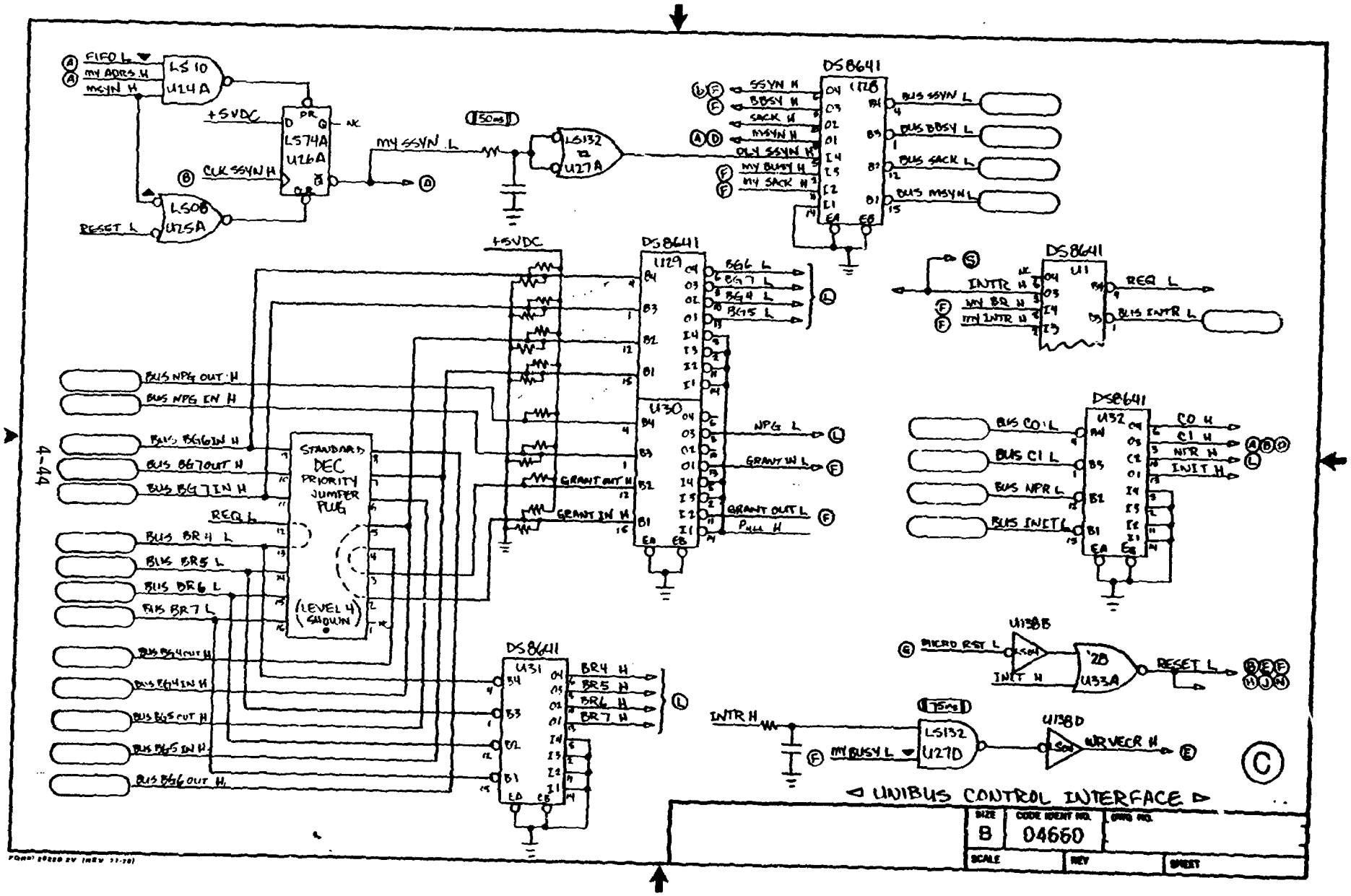
During an association cycle, the assertion of DATA CYC H negates BUF RFSH H, and the "RAS" signals are again generated within the row address window. With both input-select signals LOW, the multiplexers select the "1CO" and "2CO" inputs, which are the low bytes of the source for the association. The ARAM and DRAM sources were described in the previous paragragh. On Sheet (R) the source for the CRAM association is a combination of host system status signals latched by U185 ad U186 (SN74LS378) and outputs from the ARAM, AM DO8-11 H, and DRAM, AM DO4-05 H. Approximately 25 nanoseconds after the appropriate

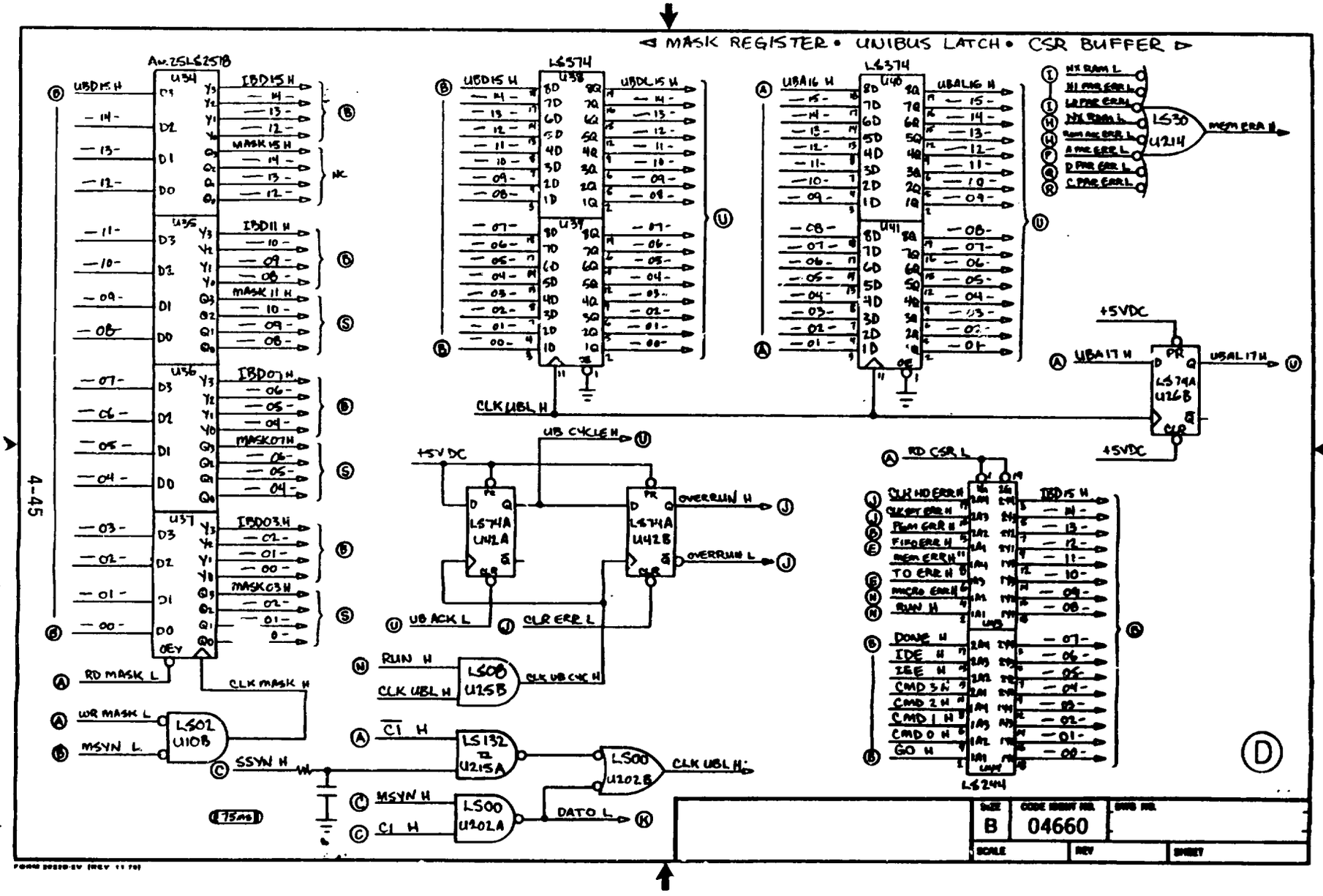
"RAS" signal is asserted, U213 is clocked into its inactive state, negating AD ROW L and C ROW L. The multiplexers now select the "1C1" and "2C1" inputs, which are the high bytes of the source for the association. After the address has stabilized, the appropriate "CAS" signal is asserted and data in the dynamic RAM is accessed. At the end of each access cycle within the association cycle, either AD STROBE H or C STROBE H will clock the data into the Associative Memory Data Register, U169 and U183 (SN74S174) or U199 (SN74S175). U169, U183, and U199 store the data, AM D00-15 H, for immediate use by the Qualifier Logic discussed in the following paragraph. In the "HALT" mode these registers are cleared by HALT L to prevent premature interaction with the Qualifier Logic. When the "CAS" signals are negated, the appropriate parity-error latches, (SN74S74), U172 and U196A, are also clocked.

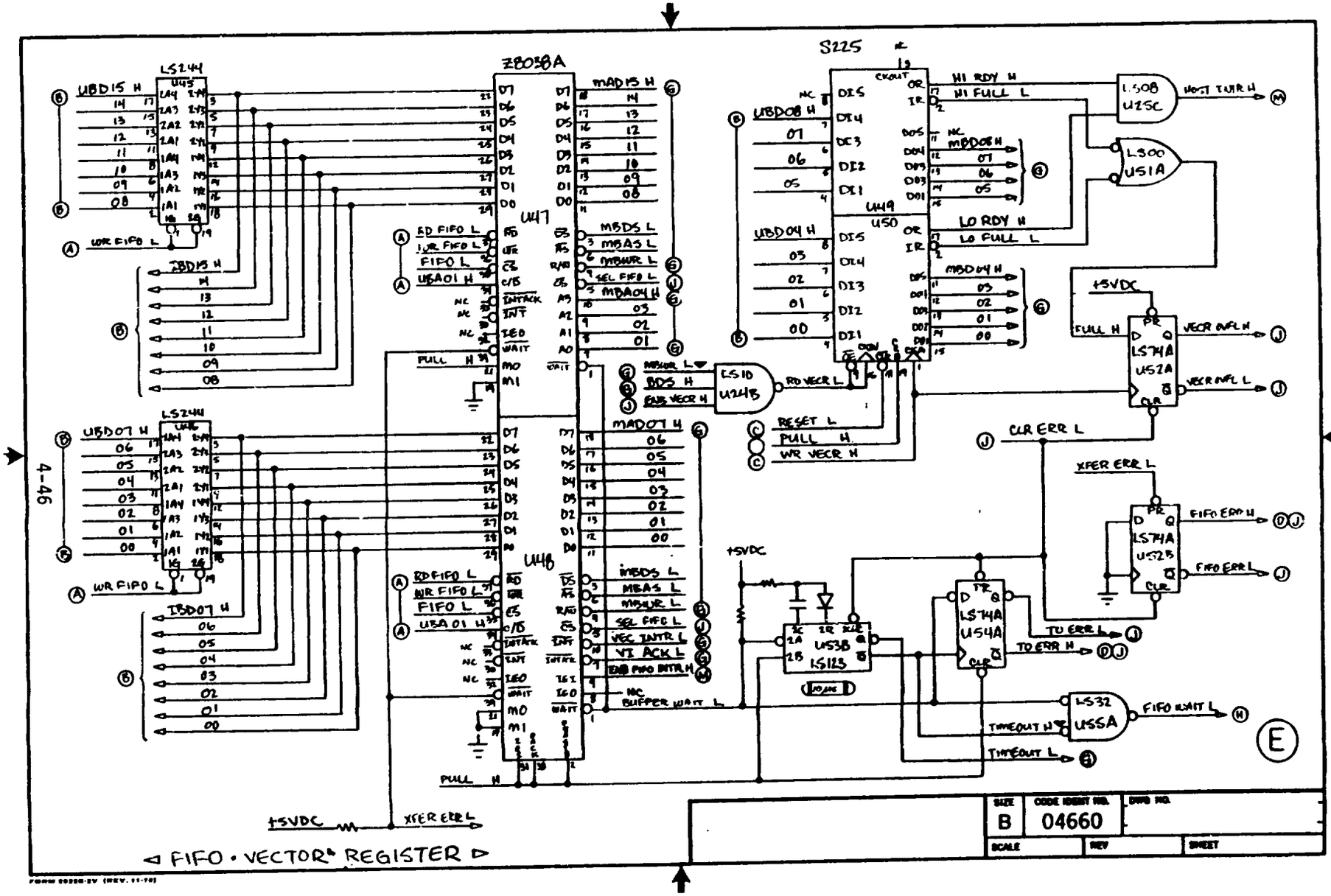

DS8641
U1
U2
U3
U4
U5
BUS A17 L
16
15
14
13
12
11
10
09
08
07
06
05
04
03
02
01
00
UBA17 H
UBA15 H
UBA11 H
UBA07 H
UBA05 H
PULL H
+5VDC
Am25LS2521
U6
U7
SW1
RP1
UBA12 H
SWA12 H
UBA11 H
SWA11 H
UBA10 H
SWA10 H
UBA09H
SWA09H
UBA08 H
SWA08 H
UBA07H
SWA07H
UBA06H
SWA06H
UBA05H
SWA05H
UBA04H
SWA04H
UBA03H
SWA03H
LS139
C1 H
UBA01H
WR MASK L
WR CSR L
RD MASK L
RD CSR L
UBA02H
MY ADRS L
LS32
U9A
LS08
U12A
U11A
MY ADRS H
FIFO H
FIFO L
U11B
MSYN H
LS20
U13A
U11C
LS74A
U14A
WR FIFO L
MY SSYN L
U13B
U14B
RD FIFO L
MSYN H
RECOVER L
UNIBUS ADDRESS DECODE
SIZE
B
04660
SCALE
REV
SHEET

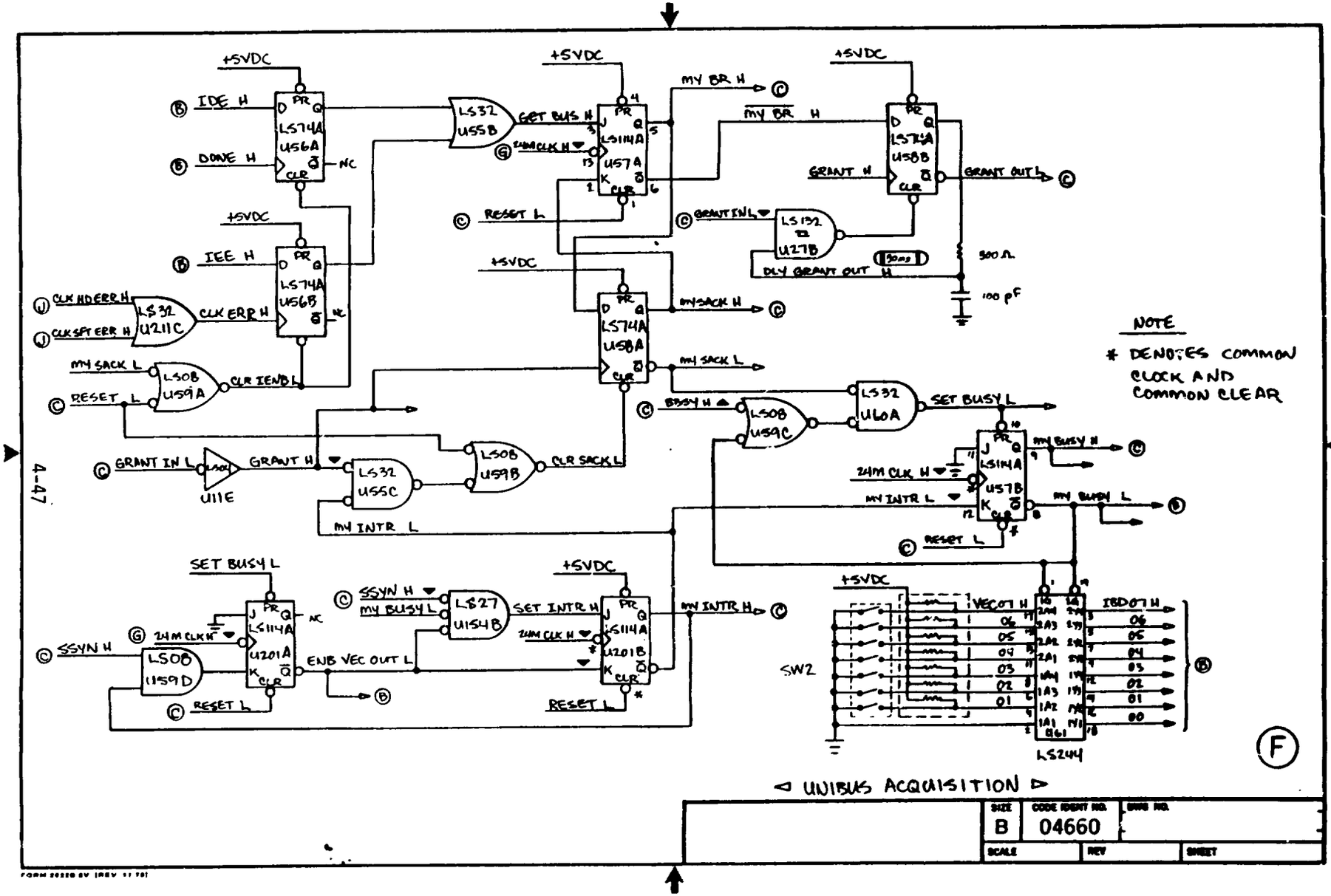

DS8641
U15
BUS DI5 L
14
13
12
BUS DI1 L
11
10
09
08
BUS DI7 L
06
05
04
BUS DI3 L
03
02
01
00
U16
U17
U18
ENB HI L
ENB LO L
ENB VEC OUT L
MY SSYN L
CI H
LS32 U19D
LS08 U12B
UBD15 H
UBD11 H
UBD07 H
UBD03 H
IBD15 H
IBD11 H
IBD07 H
IBD03 H
I/O RD L
ENB CSR L
I/O WR L
LS32 U9B
LS32 U9C
MRD CSR L
RESET L
MBD07 H
LS74A U19A
DONE H
MAR CSR L
GO L
Am25LS2518
MBD04 H
MBD00 H
CMD03 H
CMD00 H
U20
+5VDC
UBD00 H
LS74A U21A
GO H
UBD06 H
IDE H
UBD05 H
IEE H
U21B
MSYN H
LS04 U11D
MSYN L
WR CSR L
LS02 U10A
CLK CSR H
LS08 U12D
WR FIFO L
RD FIFO L
LS08 U12C
LD RECOVER L
LS195A U22
GM CLK H
RECOVER H
U23A
U23B
CLK SSYN H
RECOVER L
RAM ACC ERR L
PGM ERR H
ITM ERR L
CLR ERR L
LS74A U10B
UNIBUS DATA • INTERNAL DATA BUS • CONTROL REGISTER
B
04660
SIZE
SCALE
REV
SHEET
4-42

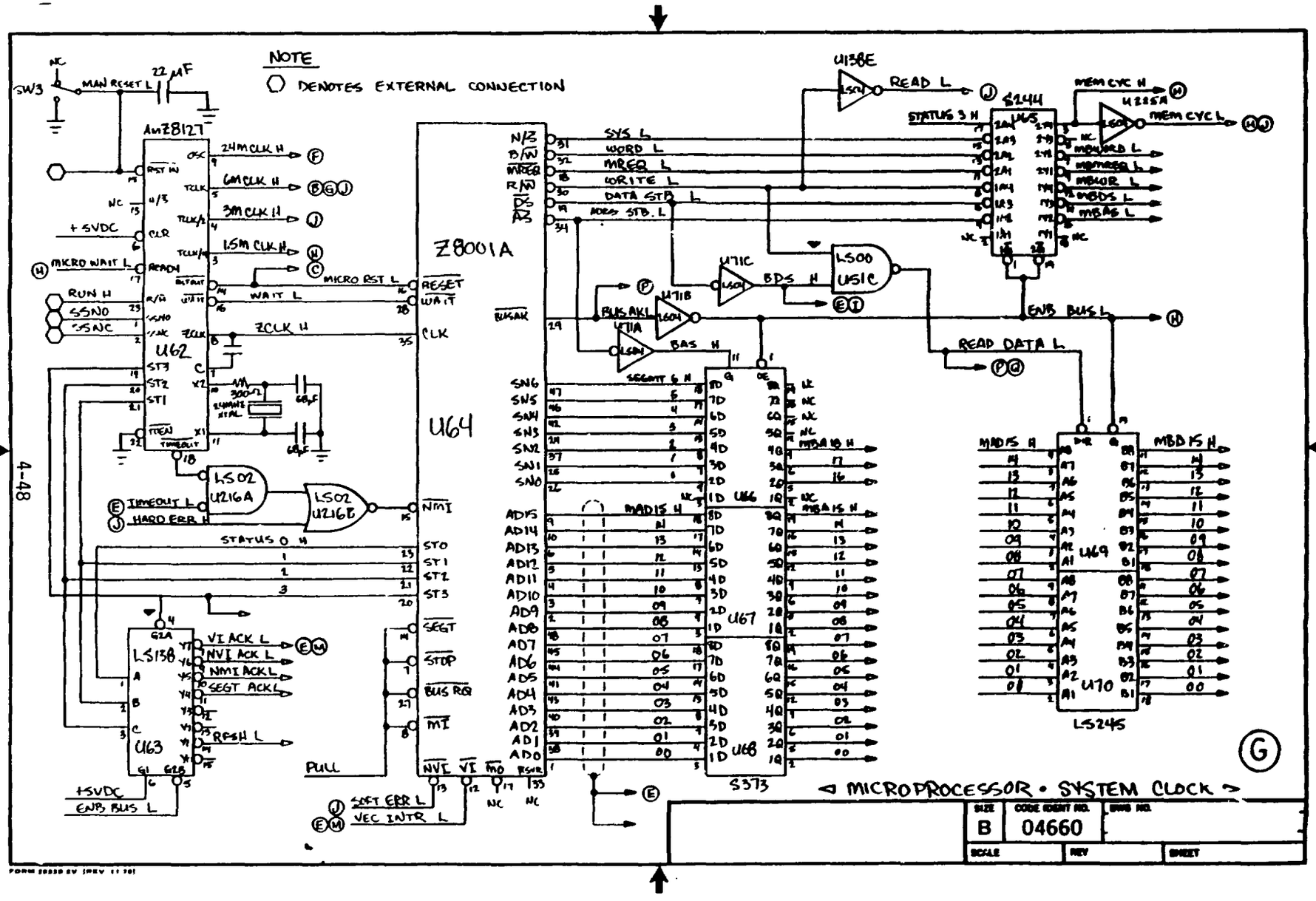

DS8641
U15
U16
U17
U18
BUS D15 L
BUS D07 L
UBD15 H
UBD11 H
UBD07 H
UBD03 H
IBD15 H
IBD11 H
IBD07 H
IBD03 H
ENB H1 L
ENB LO L
ENB VEC OUT !
MSYN H
MSYN L
WR CSR L
RD CSR L
I/O RD L
I/O WR L
ENB CSR L
RESET L
GO L
GO H
CLK CSR H
WR FIFO L
RD FIFO L
LD RECOVER L
RECOVER H
GM CLK H
CLK SYN H
RAM ACC ERR L
PGM ERR H
PGM ERR L
PLR ERR L
IDE H
IEE H
DONE H
MAR CSR L
LS32 U9B
LS32 U9C
LS32 U9D
LS04 U11D
LS02 U10A
LS08 U12D
LS08 U12C
LS08 U12B
LS74A U9A
LS74A U10A
LS74A U10B
LS74A U21A
LS74A U21B
LS74A U22
LS74A U23A
LS74A U23B
Am25LS2518
U20
+5VDC
◁ UNIBUS DATA • INTERNAL DATA BUS • CONTROL REGISTER ▷
B
0456U
SIZE
SCALE
REV
SHEET

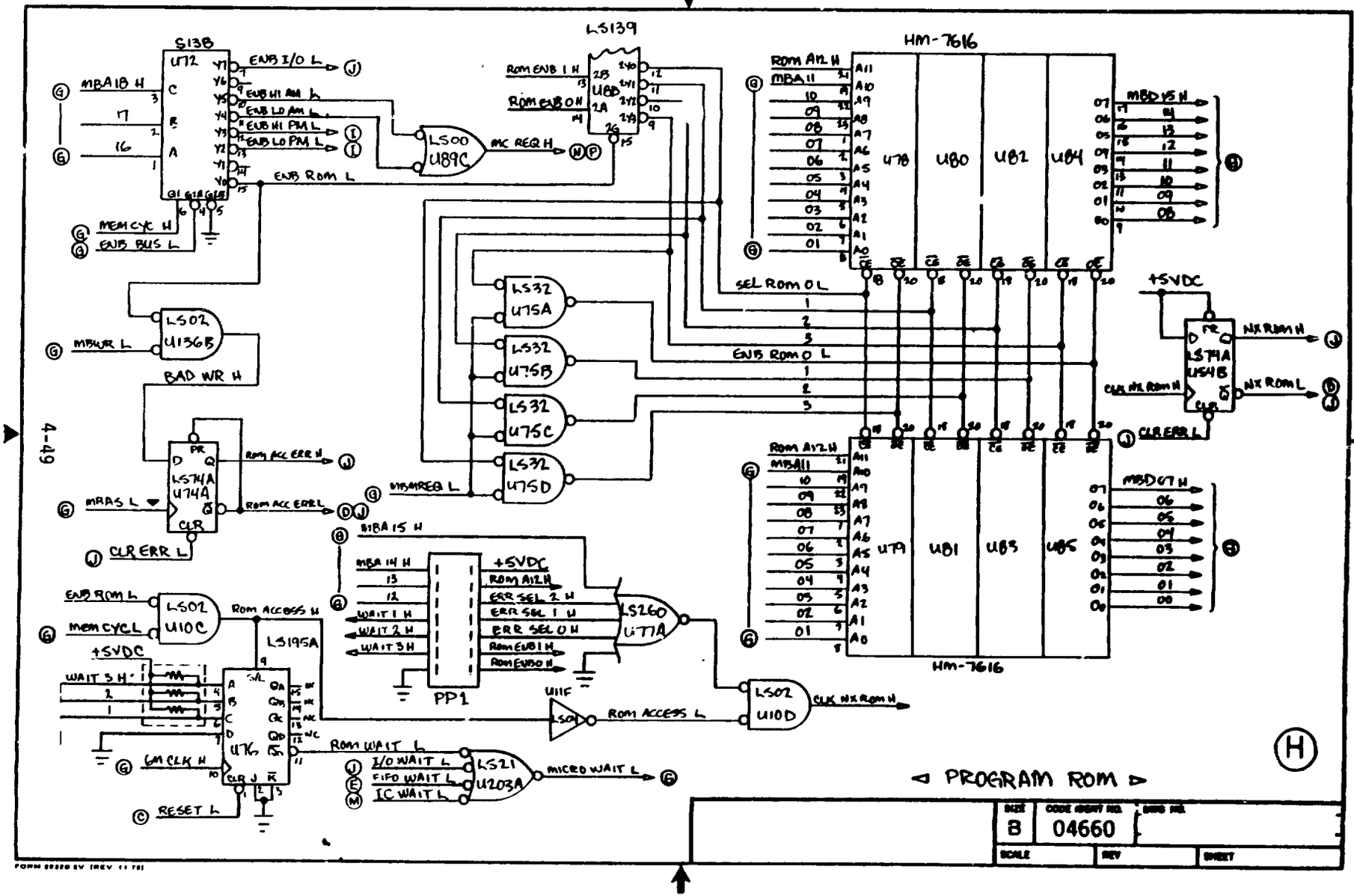

FIFO L
MY ADRS H
MSYN H
LS10 U14A
+5VDC
LS74A U26A
CLK SSYN H
RESET L
LS08 U25A
MY SSYN L
50ns
LS132 U27A
DS8641 U128
SSYN H
BBSY H
SACK H
MSYN H
OWN SSYN H
MY BUSY H
MY SACK H
BUS SSYN L
BUS BBSY L
BUS SACK L
BUS MSYN L
DS8641 U129
BG6 L
BG7 L
BG4 L
BG5 L
DS8641 U1
INTR H
MY RQ H
MY INTR H
REQ L
BUS INTR L
BUS NPG OUT H
BUS NPG EN H
BUS BG6 IN H
BUS BG7 OUT H
BUS BG7 IN H
REQ L
BUS BR4 L
BUS BR5 L
BUS BR6 L
BUS BR7 L
STANDARD DEC PRIORITY JUMPER PLUG
(LEVEL 4 SHOWN)
DS8641 U130
NPG L
GRANT IN L
GRANT IN H
GRANT OUT H
GRANT OUT L
PULL H
DS8641 U131
BR4 H
BR5 H
BR6 H
BR7 H
DS8641 U132
BUS CO L
BUS CI L
BUS NPR L
BUS INIT L
CO H
CI H
NTR H
INIT H
MICRO RST L
INIT H
U138B
'20 U33A
RESET L
INTR H
MY BUSY L
75ns
LS132 U27D
U138D
VECT H
UNIBUS CONTROL INTERFACE
SIZE B
CODE IDENT NO. 04660
SCALE
REV
SHEET
4-44

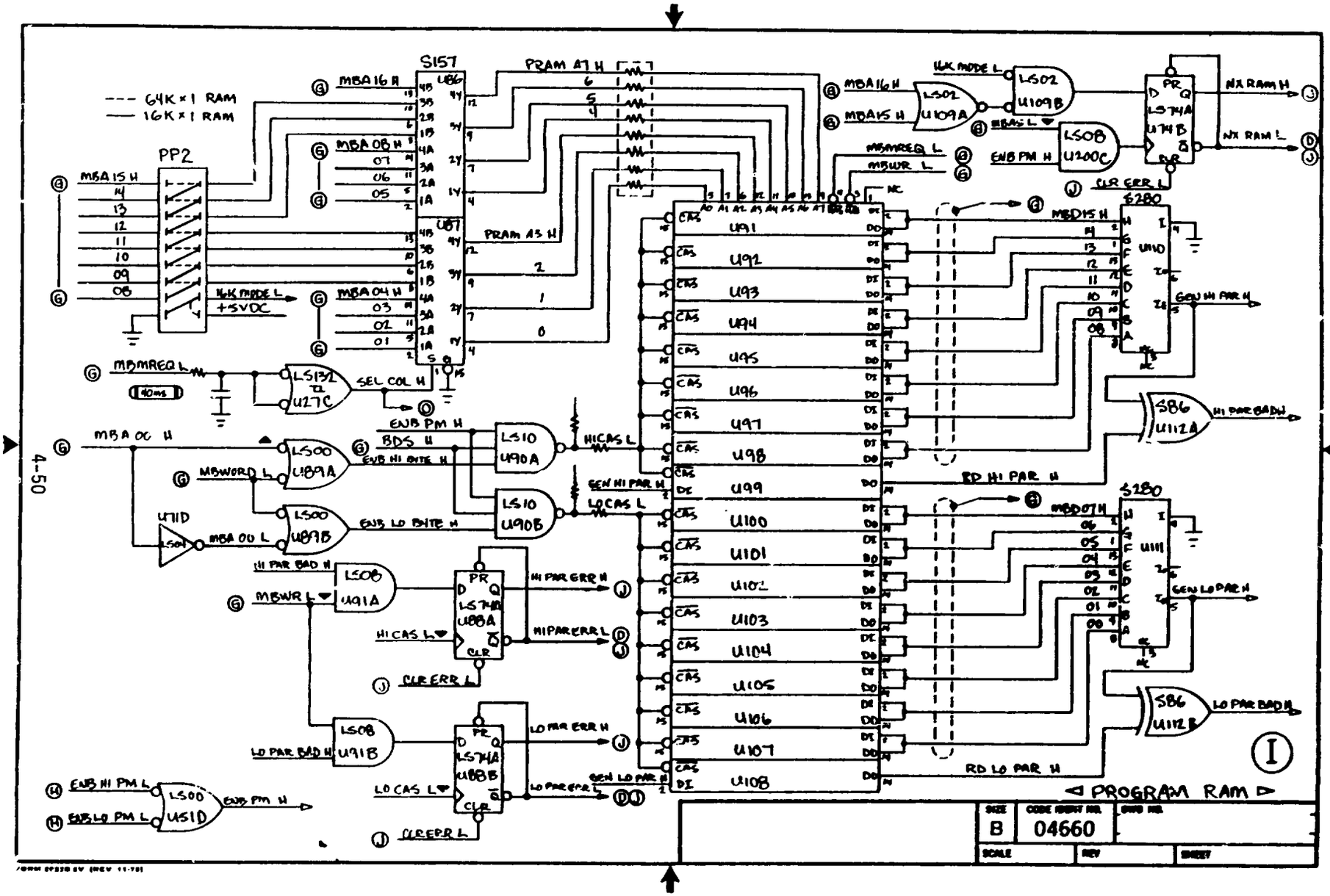

MASK REGISTER • UNIBUS LATCH • CSR BUFFER
Am25LS2518
U34
U35
U36
U37
IBD15 H
IBD11 H
IBD07 H
IBD03 H
MASK15 H
MASK11 H
MASK07 H
MASK03 H
RD MASK L
CLK MASK H
WR MASK L
MSYN L
SSYN H
LS02 U10B
LS374
U38
U39
U40
U41
UBD15 H
UBA16 H
UBDL15 H
UBAL16 H
CLK UBL H
LS30 U21H
MEM ERR H
+5VDC
UBA17 H
UBAL17 H
LS74A U26B
UB CYCLE H
LS74A U41A
LS74A U42B
OVERRUN H
OVERRUN L
UB ACK L
CLR ERR L
RUN H
LS08 U15B
CLK UB CYC H
C1 H
LS132 U21SA
LS00 U20ZB
MSYN H
LS00 U20ZA
DATO L
RD CSR L
CLK HD ERR H
TIMEOUT ERR H
PARITY ERR H
FIFO ERR H
MEM ERR H
TO ERR H
MICRO ERR H
RUN H
DONE H
IDE H
IEE H
CMD 3 H
CMD 2 H
CMD 1 H
CMD 0 H
GO H
LS244
175ns
4-45
D
SIZE
B
CODE IDENT NO.
04660
DWG NO.
SCALE
REV
SHEET

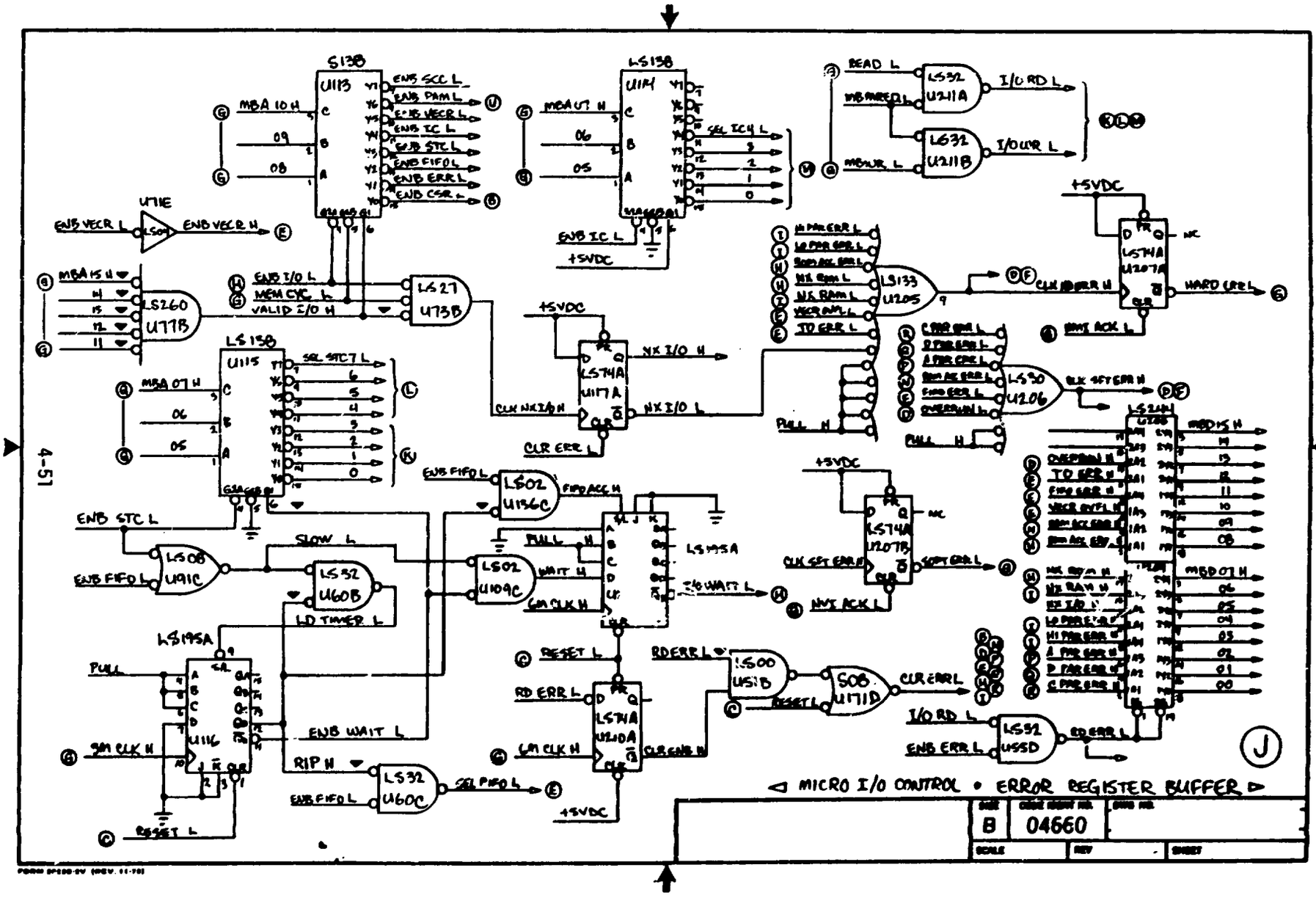

LS244
U45
UBD15 H
WR FIFO L
IBD15 H
LS244
U46
UBD07 H
WR FIFO L
IBD07 H
Z8038A
U47
U48
RD FIFO L
WR FIFO L
FIFO L
UBA01 H
MBDS L
MBAS L
MBWR L
SEL FIFO L
MBA04 H
MBD15 H
MBD07 H
PULL H
+5VDC
XFER ERR L
S225
U49
U50
UBD08 H
UBD04 H
HI RDY H
HI FULL L
LO RDY H
LO FULL L
MBD08 H
MBD04 H
LS08
U25C
LS00
U51A
HOST INTR H
LS10
U24B
RD VECR L
RESET L
WR VECR H
LS74A
U52A
FULL H
VECR OVFL H
VECR OVFL L
CLR ERR L
XFER ERR L
LS74A
U52B
FIFO ERR H
FIFO ERR L
LS74A
U54A
TO ERR L
TO ERR H
LS123
U53B
LS32
U55A
FIFO WAIT L
TIMEOUT H
TIMEOUT L
FIFO • VECTOR • REGISTER
SIZE
B
CODE IDENT NO.
04660
DWG NO.
SCALE
REV
SHEET

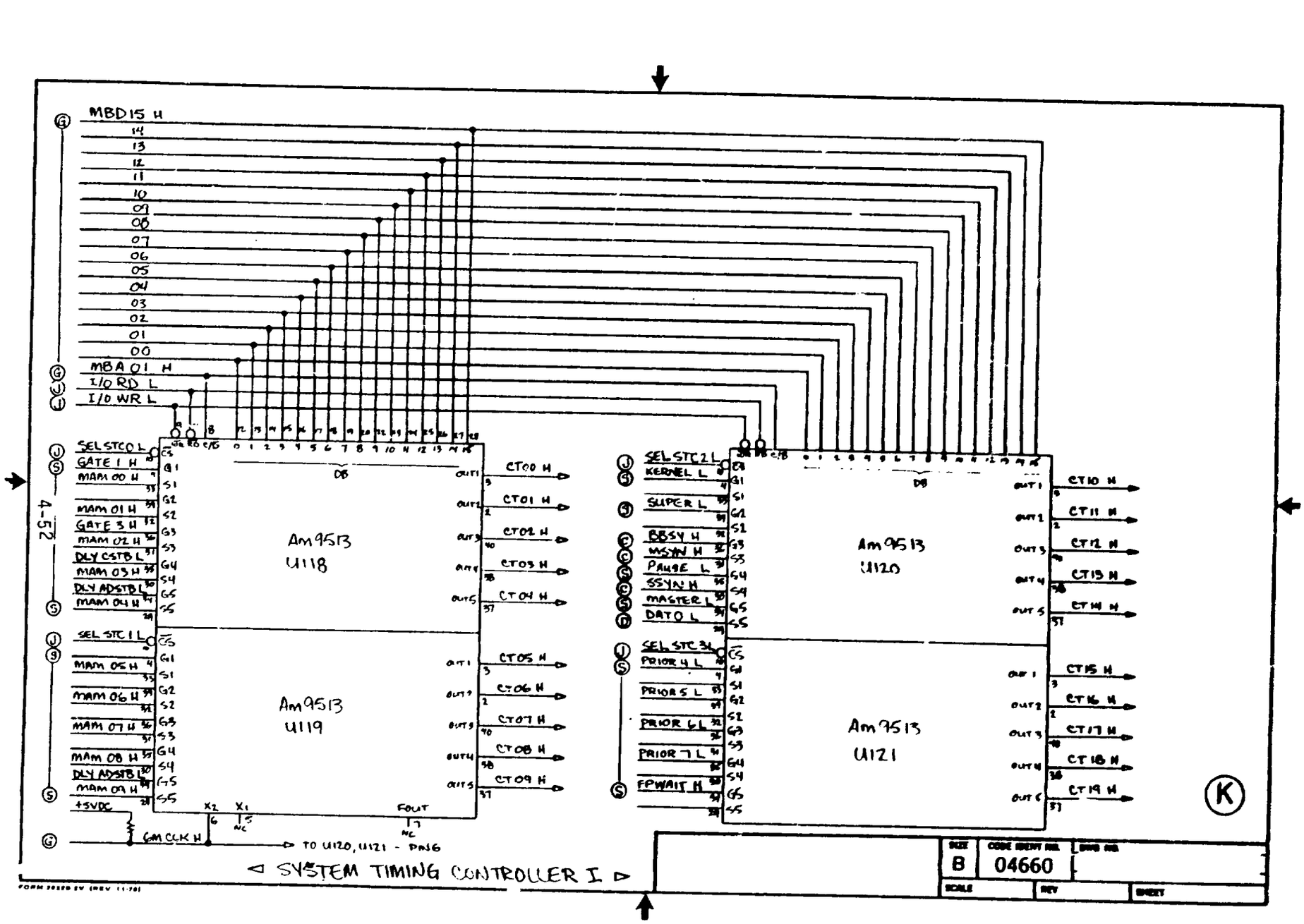

+5VDC
IDE H
DONE H
LS74A U56A
NC
LS32 U55B
GET BUS H
24MCLK H
LS114A U57A
MY BR H
LS74A U58B
GRANT H
GRANT OUT L
RESET L
GRANT IN L
LS132 U27B
DLY GRANT OUT H
50ns
500 Ω
100 pF
IEE H
LS74A U56B
CLK HDERR H
CLK SFT ERR H
LS32 U21IC
CLK ERR H
MY SACK L
LS08 U59A
CLR IENBL
LS74A U58A
MY SACK H
BSSY H
LS08 U59C
LS32 U6A
SET BUSY L
MY BUSY H
LS114A U57B
MY INTR L
MY BUSY L
GRANT IN L
U11E
LS32 U55C
LS08 U59B
CLR SACK L
SET BUSY L
SSYN H
MY BUSY L
LS27 U15 4B
SET INTR H
LS114A U201A
LS08 U159D
ENB VEC OUT L
LS114A U201B
MY INTR H
RESET L
SW2
VEC07 H
IBD07 H
LS244
NOTE
* DENOTES COMMON CLOCK AND COMMON CLEAR
4-47
F
UNIBUS ACQUISITION
SIZE B
CODE IDENT NO. 04660
DWG NO.
SCALE
REV
SHEET

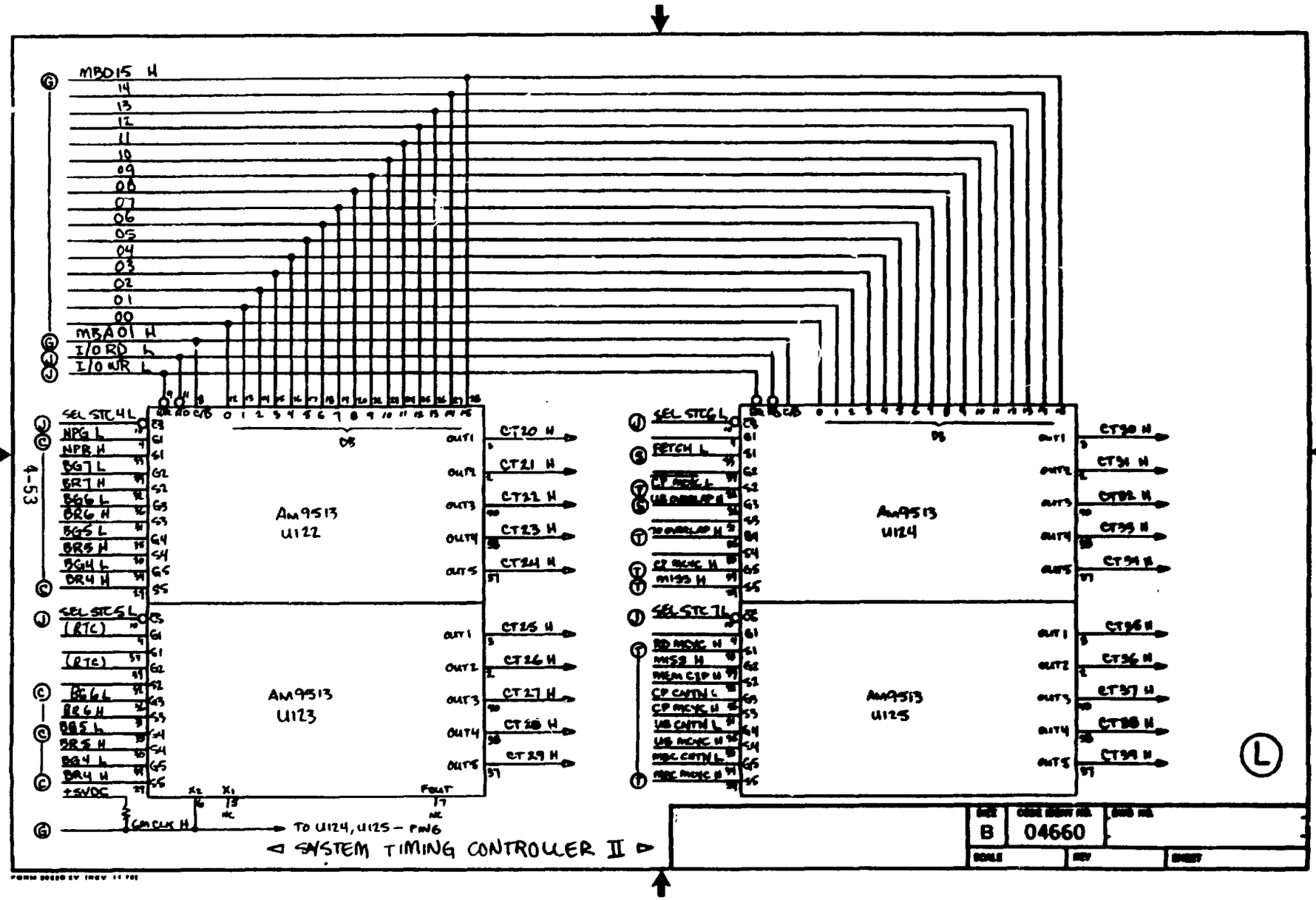

NOTE
DENOTES EXTERNAL CONJECTION
SW3
MAN RESET L
22 µF
AmZ8127
U62
24M CLK H
6M CLK H
3M CLK H
1.5M CLK H
+5VDC
MICRO WAIT L
RUN H
MICRO RST L
WAIT L
ZCLK H
300Ω
ELEMENT XTAL
Z8001A
U64
RESET
WAIT
CLK
NMI
STO
ST1
ST2
ST3
SEGT
STOP
BUS RQ
MI
PULL
NVI
VI
MO
RSHR
SYS L
WORD L
MREQ L
WRITE L
DATA STB L
ADRS STB L
BUSAK
BUS AK L
BAS H
SEGMT 6 H
SN6
SN5
SN4
SN3
SN2
SN1
SN0
AD15
AD14
AD13
AD12
AD11
AD10
AD9
AD8
AD7
AD6
AD5
AD4
AD3
AD2
AD1
AD0
MAD15 H
U66
U67
U68
S373
LS138
U63
G2A
G2B
G1
VI ACK L
NVI ACK L
NMI ACK L
SEGT ACK L
RESH L
+5VDC
ENB BUS L
TIMEOUT L
HARD ERR H
LS02
U216A
U216B
STATUS 0 H
STATUS 3 H
SOFT ERR L
VEC INTR L
U138E
READ L
LS244
U65
MEM CYC H
U228A
MEM CYC L
MBWORD L
MBMREQ L
MBIOR L
MBDS L
MBAS L
U71C
U71B
U71A
LS00
U51C
BDS H
ENB BUS L
READ DATA L
MBD15 H
U69
U70
LS245
MICROPROCESSOR • SYSTEM CLOCK
SIZE
B
CODE IDENT NO.
04660
DWG NO.
SCALE
REV
SHEET
4-48

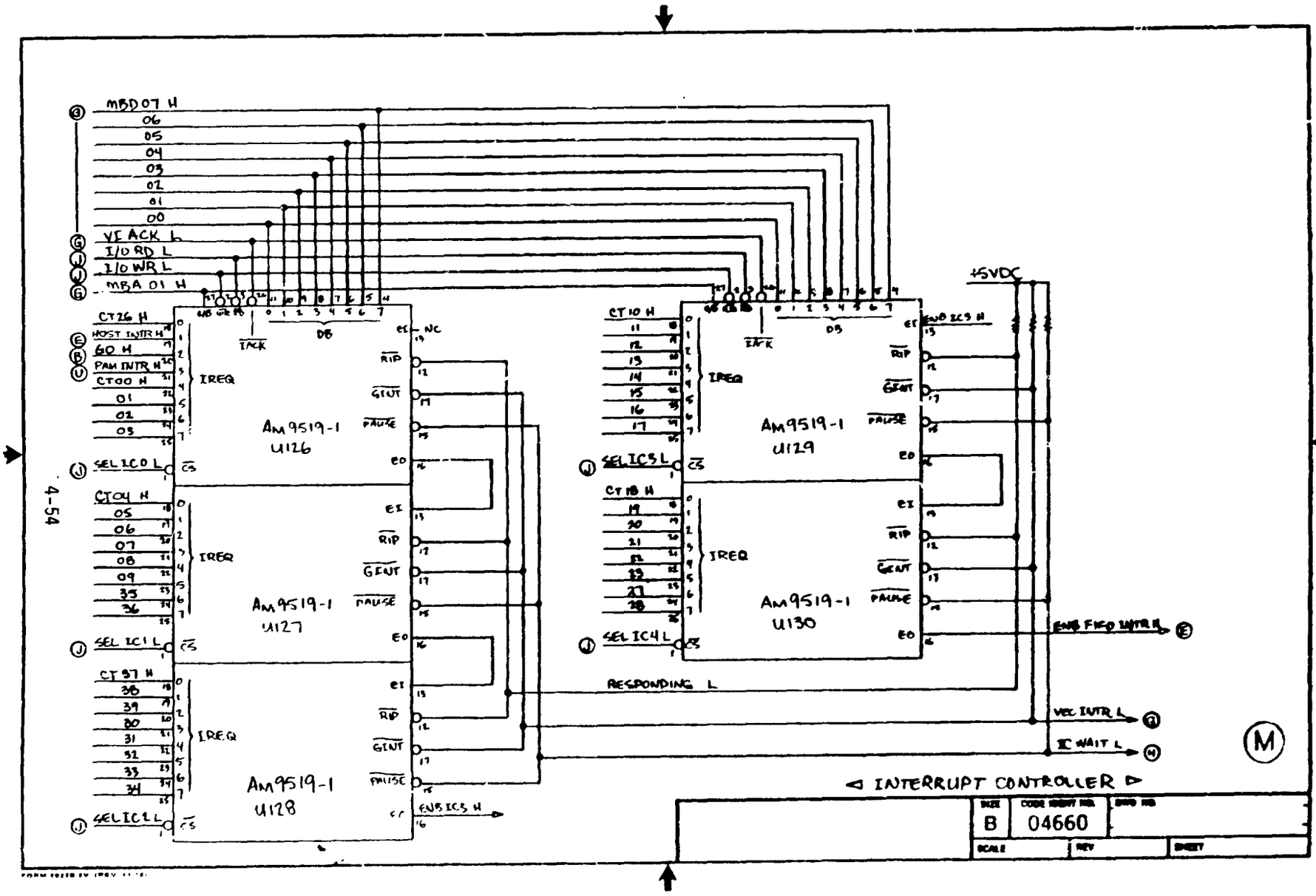

S138
U72
MBA18 H
MBA17
MBA16
ENB I/O L
ENB HI AM L
ENB LO AM L
ENB HI PM L
ENB LO PM L
ENB ROM L
MEM CYC H
ENB BUS L
LS00
U89C
MC REQ H
LS139
U8D
ROM ENB 1 H
ROM ENB 0 H
HM-7616
ROM A11 H
MBA11
MBD15 H
U78
U80
U82
U84
SEL ROM 0 L
LS32
U75A
U75B
U75C
U75D
ENB ROM 0 L
+5VDC
LS74A
U54B
NX ROM H
NX ROM L
CLK NX ROM H
CLR ERR L
LS02
U136B
MBWR L
BAD WR H
LS74A
U74A
ROM ACC ERR H
ROM ACC ERR L
MRAS L
DMAREQ L
ROM A12 H
MBA11
MBD07 H
U79
U81
U83
U85
HM-7616
BIBA 15 H
MBA 14 H
WAIT 1 H
WAIT 2 H
WAIT 3 H
ROM ENB 1 H
ROM ENB 0 H
PP1
LS260
U77A
ENB ROM L
MEM CYC L
LS02
U110C
ROM ACCESS H
LS195A
U76
WAIT 3 H
MA CLK H
RESET L
ROM WAIT L
I/O WAIT L
FIFO WAIT L
IC WAIT L
LS21
U203A
MICRO WAIT L
U111F
ROM ACCESS L
LS02
U110D
CLK NX ROM H
PROGRAM ROM
SIZE
B
CODE IDENT NO.
04660
DWG NO.
SCALE
REV
SHEET
4-49
H

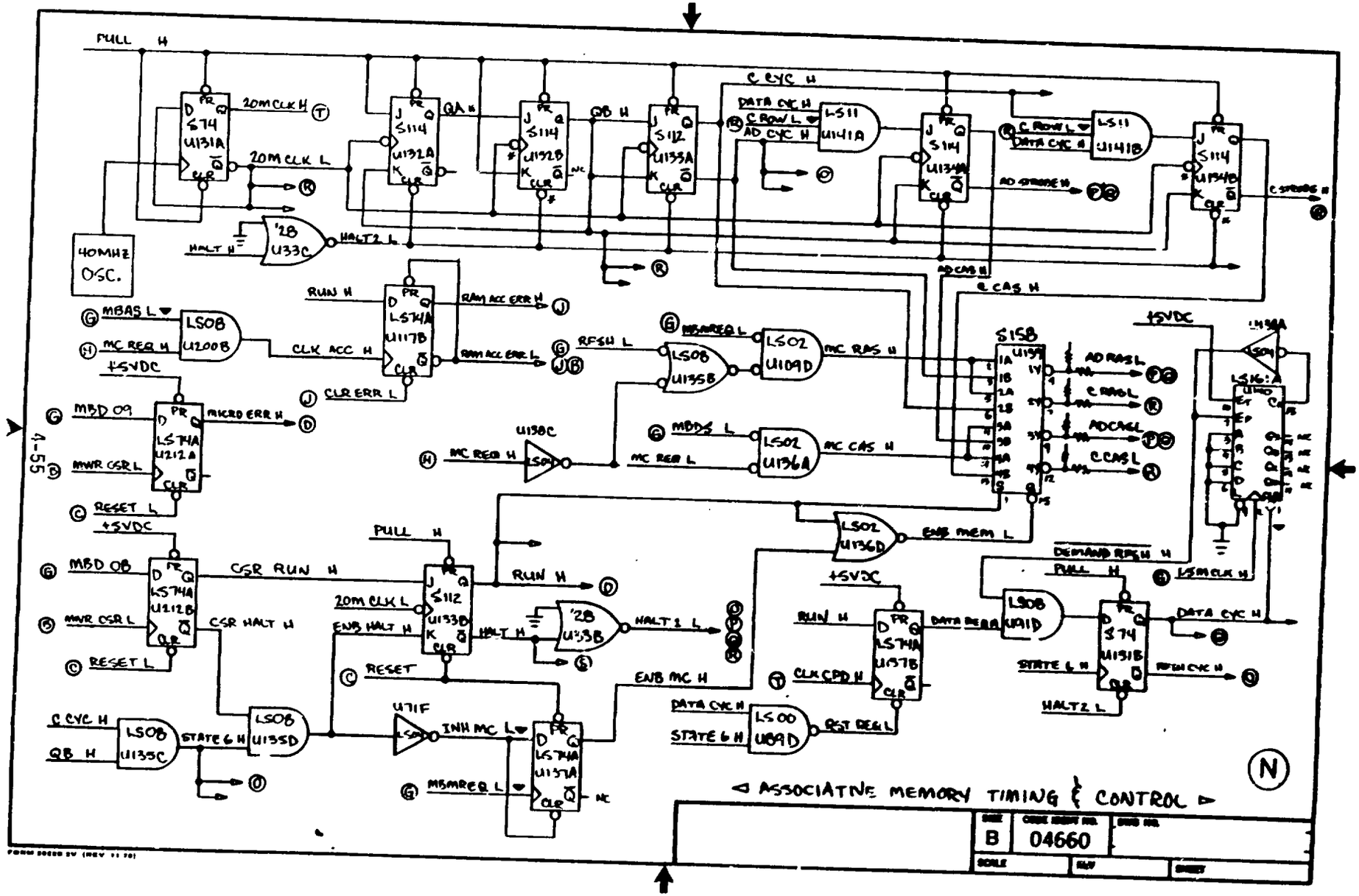

--- 64K x 1 RAM
—— 16K x 1 RAM
PP2
S157
U86
U87
MBA 15 H
MBA 14
MBA 13
MBA 12
MBA 11
MBA 10
MBA 09
MBA 08
MBA 08 H
MBA 07
MBA 06
MBA 05
MBA 04 H
MBA 03
MBA 02
MBA 01
16K MODE L
+5VDC
PRAM A7 H
PRAM A5 H
2
1
0
MBMREQ L
40ns
LS132 U27C
SEL COL H
ENB PM H
MBA 00 H
BDS H
ENB HI BYTE H
MBWORD L
LS00 U89A
LS00 U89B
U11D
MBA 00 L
ENB LO BYTE H
LS10 U90A
LS10 U90B
HICAS L
LOCAS L
GEN HI PAR H
GEN LO PAR H
HI PAR BAD H
MBWR L
LS08 U91A
LS74A U88A
HI PAR ERR H
HIPARERR L
HI CAS L
CLR ERR L
LO PAR BAD H
LS08 U91B
LS74A U88B
LO PAR ERR H
LO PARERR L
LO CAS L
ENB HI PM L
ENB LO PM L
LS00 U51D
ENB PM H
U91
U92
U93
U94
U95
U96
U97
U98
U99
U100
U101
U102
U103
U104
U105
U106
U107
U108
MBA16 H
MBA15 H
16K MODE L
LS02 U109A
LS02 U109B
LS08 U200C
MBAS L
MBMREQ L
MBWR L
ENB PM H
LS74A U74B
NX RAM H
NX RAM L
CLR ERR L
$280 U110
$280 U111
MBDI15 H
MBDI07 H
GEN HI PAR H
GEN LO PAR H
S86 U112A
S86 U112B
HI PAR BAD H
LO PAR BAD H
RD HI PAR H
RD LO PAR H
PROGRAM RAM
I
SIZE B
CODE IDENT NO. 04660
SCALE
REV
SHEET
4-50

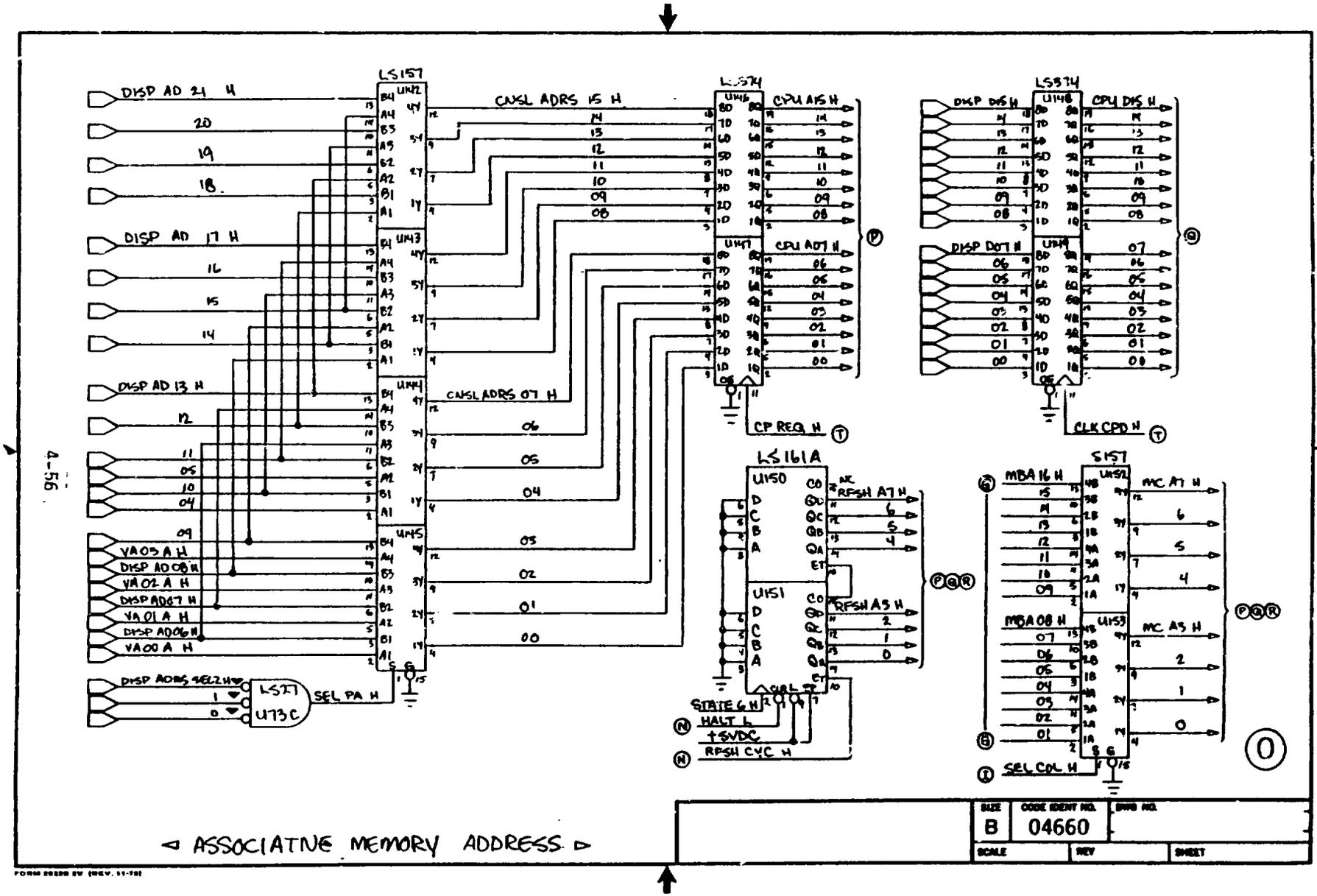

MICRO I/O CONTROL • ERROR REGISTER BUFFER
B
04660
4-51

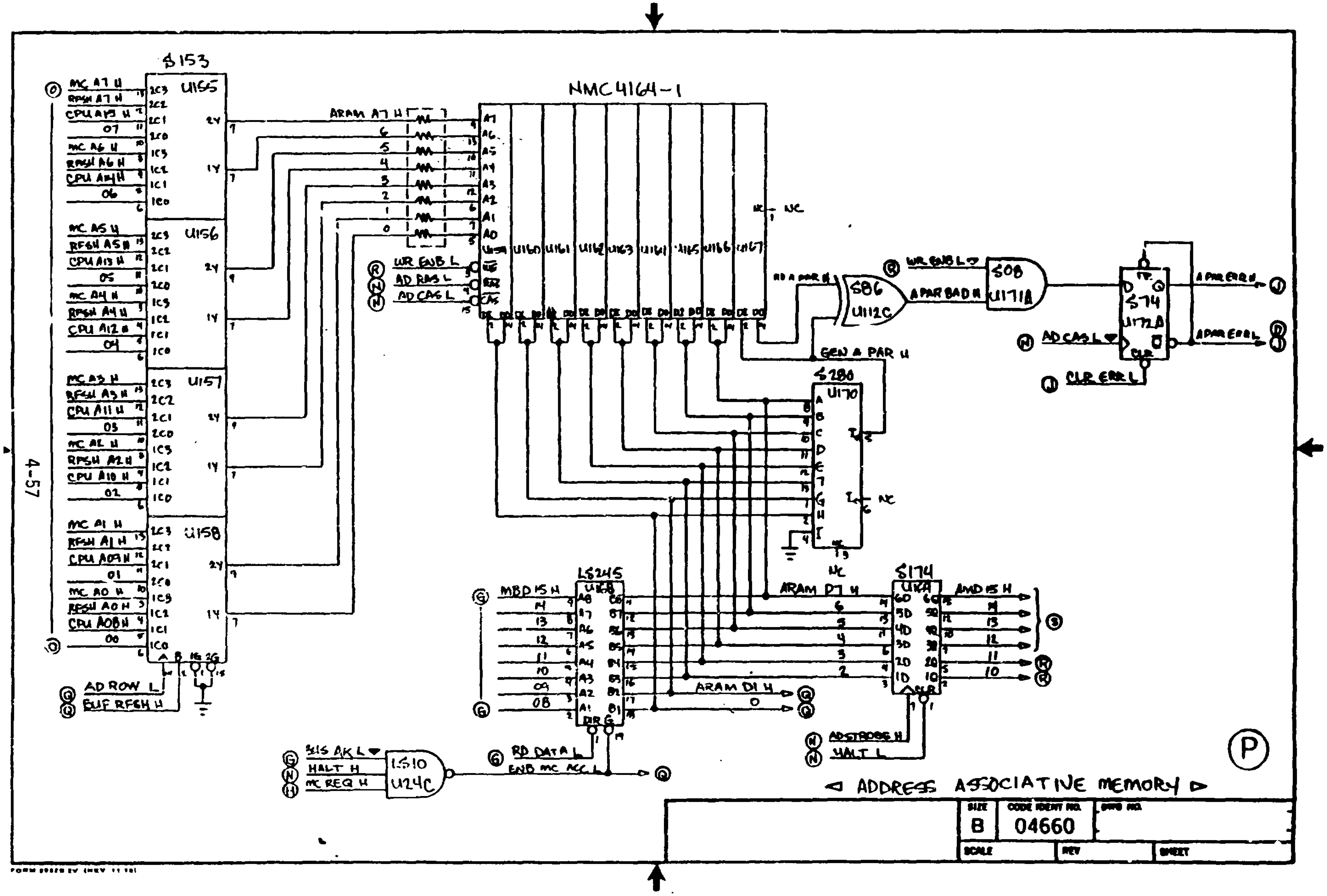

MBD15 H
14
13
12
11
10
09
08
07
06
05
04
03
02
01
00
MBA01 H
I/O RD L
I/O WR L
SEL STC0 L
GATE 1 H
MAM 00 H
MAM 01 H
GATE 3 H
MAM 02 H
DLY CSTB L
MAM 03 H
DLY ADSTB L
MAM 04 H
SEL STC1 L
MAM 05 H
MAM 06 H
MAM 07 H
MAM 08 H
DLY ADSTB L
MAM 09 H
+5VDC
GM CLK H
TO U120, U121 - PAGE
Am9513
U118
Am9513
U119
CT00 H
CT01 H
CT02 H
CT03 H
CT04 H
CT05 H
CT06 H
CT07 H
CT08 H
CT09 H
SEL STC2 L
KERNEL L
SUPER L
BBSY H
MSYN H
PAUSE L
SSYN H
MASTER L
DATO L
SEL STC3 L
PRIOR4 L
PRIOR5 L
PRIOR6 L
PRIOR7 L
FPWAIT H
Am9513
U120
Am9513
U121
CT10 H
CT11 H
CT12 H
CT13 H
CT14 H
CT15 H
CT16 H
CT17 H
CT18 H
CT19 H
SYSTEM TIMING CONTROLLER I
B
04660
4-52


MBD15 H
14
13
12
11
10
09
08
07
06
05
04
03
02
01
00
MBA01 H
I/O RD L
I/O WR L
SEL STC4 L
NPG L
NPB H
BG7 L
BR7 H
BG6 L
BR6 H
BG5 L
BR5 H
BG4 L
BR4 H
+5VDC
SEL STC5 L
(RTC)
(RTC)
BG6 L
BR6 H
BG5 L
BR5 H
BG4 L
BR4 H
Am9513
U122
Am9513
U123
CT20 H
CT21 H
CT22 H
CT23 H
CT24 H
CT25 H
CT26 H
CT27 H
CT28 H
CT29 H
SEL STC6 L
FETCH L
SEL STC7 L
RD MEMC H
MSS2 H
MEM CSP H
CP CNTN L
CP MCNC H
MB CNTN L
MB MCNC H
MEC CNTN L
MEC MCNC H
Am9513
U124
Am9513
U125
CT30 H
CT31 H
CT32 H
CT33 H
CT34 H
CT35 H
CT36 H
CT37 H
CT38 H
CT39 H
CMCLK H
TO U124, U125 – PW6
SYSTEM TIMING CONTROLLER II
B
04660
4-53

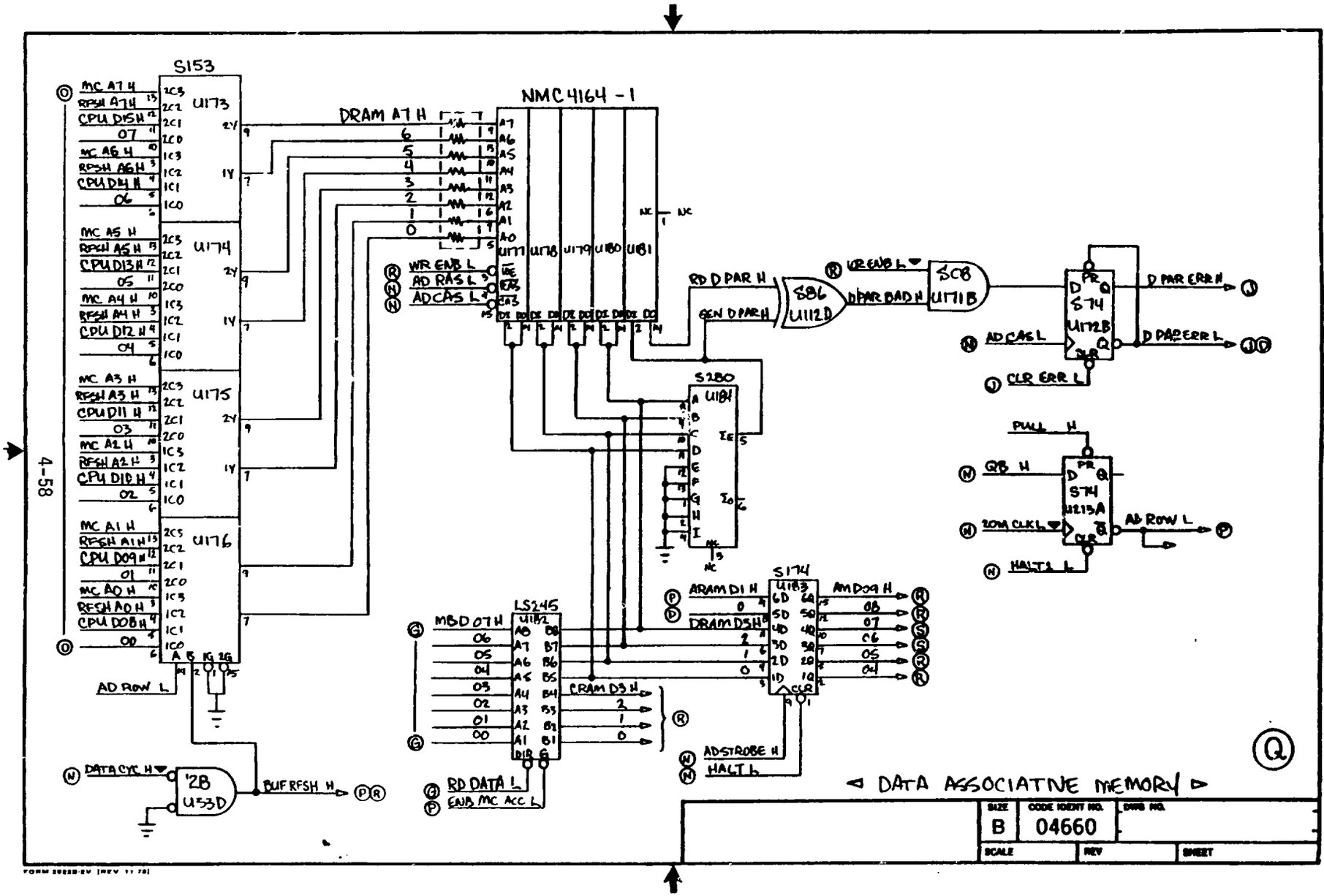

MBD 07 H
06
05
04
03
02
01
00
VE ACK L
I/O RD L
I/O WR L
MBA 01 H
+5VDC
CT26 H
HOST INTR H
60 H
PAM INTR H
CT00 H
01
02
03
SEL IC0 L
CT04 H
05
06
07
08
09
35
36
SEL IC1 L
CT 37 H
38
39
3A
31
32
33
34
SEL IC2 L
CT 10 H
11
12
13
14
15
16
17
SEL IC3 L
CT 18 H
19
20
21
22
23
27
28
SEL IC4 L
IREQ
INT
RIP
GINT
PAUSE
EO
EI
IACK
DB
AM9519-1
U126
U127
U128
U129
U130
ENB IC3 H
ENB FIFO INTR H
ENB IC2 H
RESPONDING L
VEC INTR L
IC WAIT L
INTERRUPT CONTROLLER
B
04660
SIZE
CODE IDENT NO.
DWG NO.
SCALE
REV
SHEET
4-54

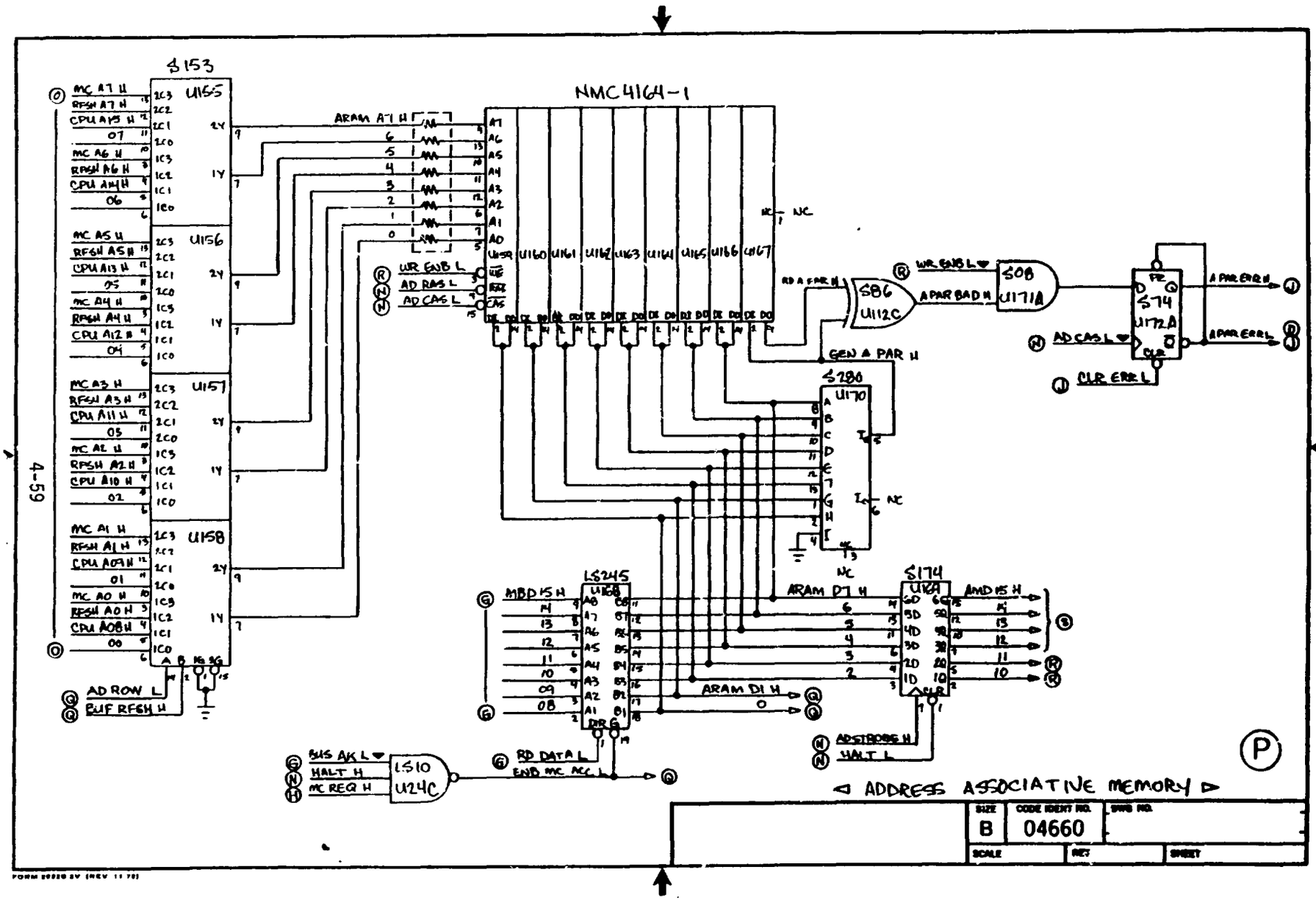

PULL H
20M CLK H
20M CLK L
40MHZ OSC.
HALT H
HALT2 L
QA H
QB H
C CYC H
DATA CYC H
C ROW L
AD CYC H
AD STROBE H
AD CAS H
C CAS H
RUN H
RAM ACC ERR H
RAM ACC ERR L
MBAS L
MC REQ H
CLK ACC H
CLR ERR L
MBD 09
MICRO ERR H
MWR CSR L
RESET L
MBD 08
CSR RUN H
CSR HALT H
ENB HALT H
RESET
INH MC L
MBMREQ L
RFSH L
MC RAS H
MBDS L
MC CAS H
ENB MEM L
AD RAS L
C RAS L
ADCAS L
CCAS L
+5VDC
DEMAND RFSH H
LSMCLK H
DATA CYC H
RFSH CYC H
STATE 6 H
CLK CPD H
DATA REQ H
RST REQ L
ASSOCIATNE MEMORY TIMING & CONTROL
B
04660

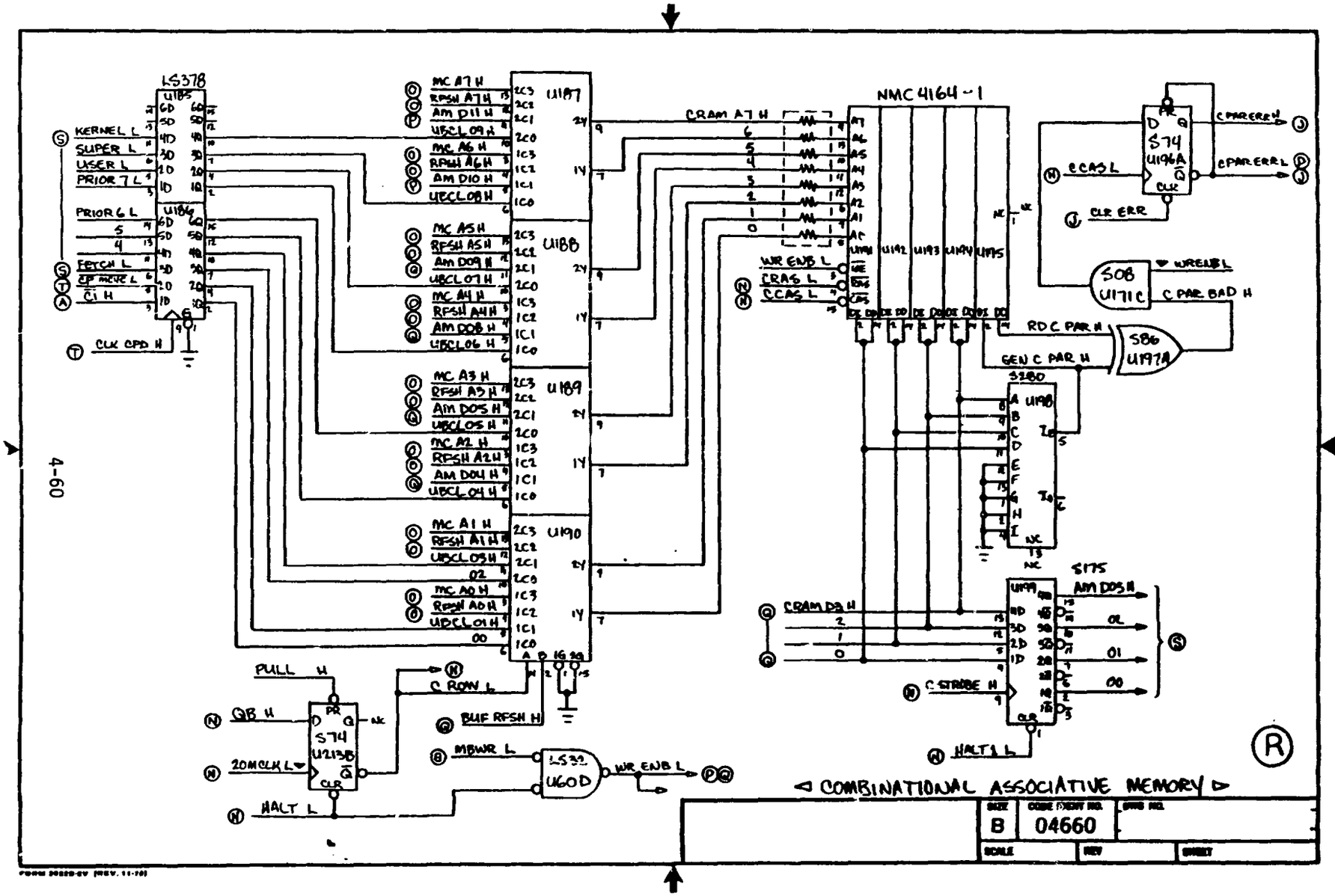

LS157
DISP AD 21 H
20
19
18
DISP AD 17 H
16
15
14
DISP AD 13 H
12
11
05
10
04
09
VA 05 A H
DISP AD 08 H
VA 02 A H
DISP AD 07 H
VA 01 A H
DISP AD 06 H
VA 00 A H
DISP ADRS SEL 2 H
LS27
U73C
SEL PA H
CNSL ADRS 15 H
CNSL ADRS 07 H
LS374
CPU A15 H
CPU A07 H
CP REQ H
DISP D15 H
DISP D07 H
CPU D15 H
CLK CPD H
LS161A
U150
U151
RFSH A7 H
RFSH A5 H
STATE 6 H
HALT L
+5VDC
RFSH CYC H
MBA 16 H
MBA 08 H
MC A7 H
MC A5 H
SEL COL H
SIZE
B
CODE IDENT NO.
04660
DWG NO.
SCALE
REV
SHEET
ASSOCIATIVE MEMORY ADDRESS

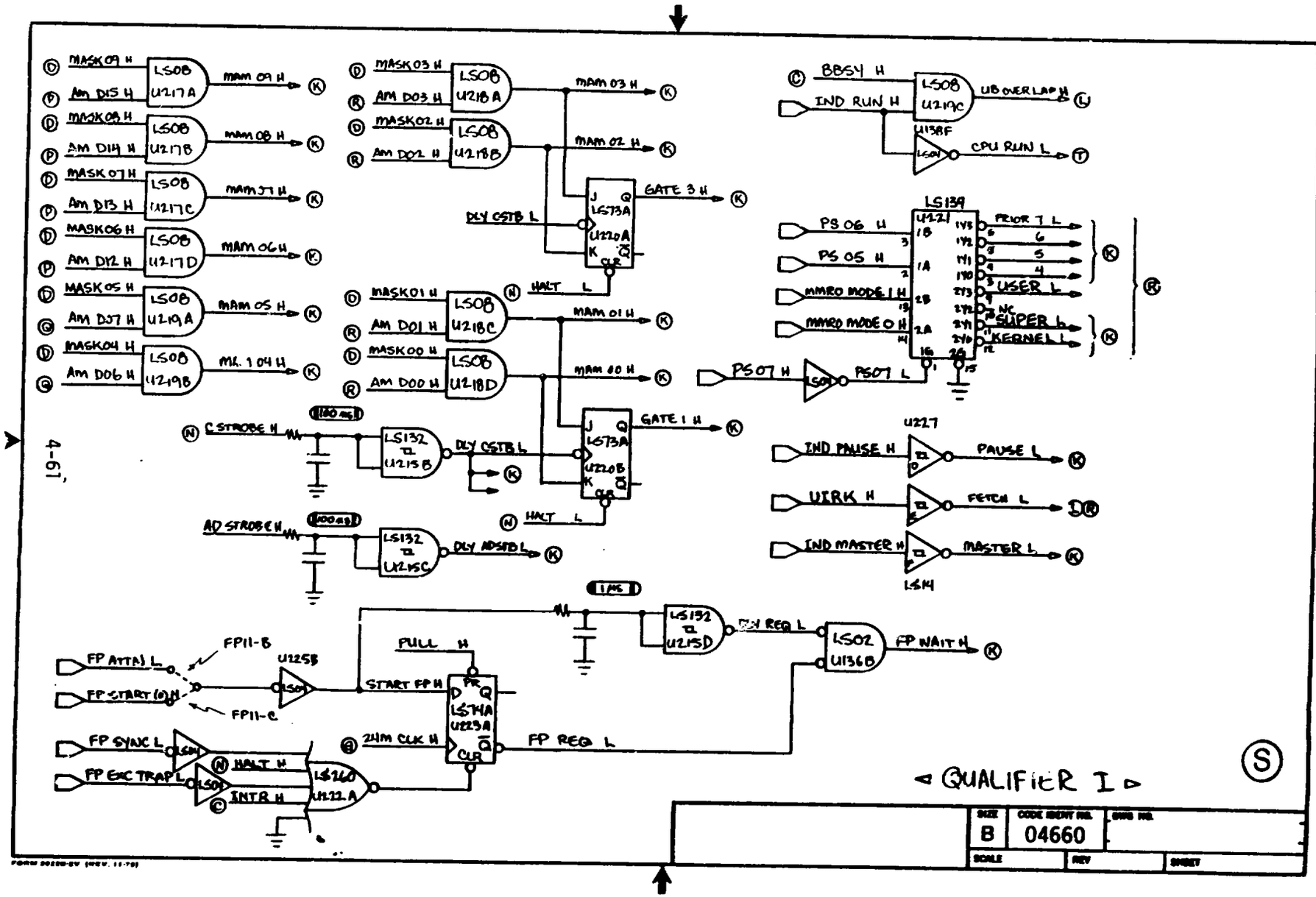


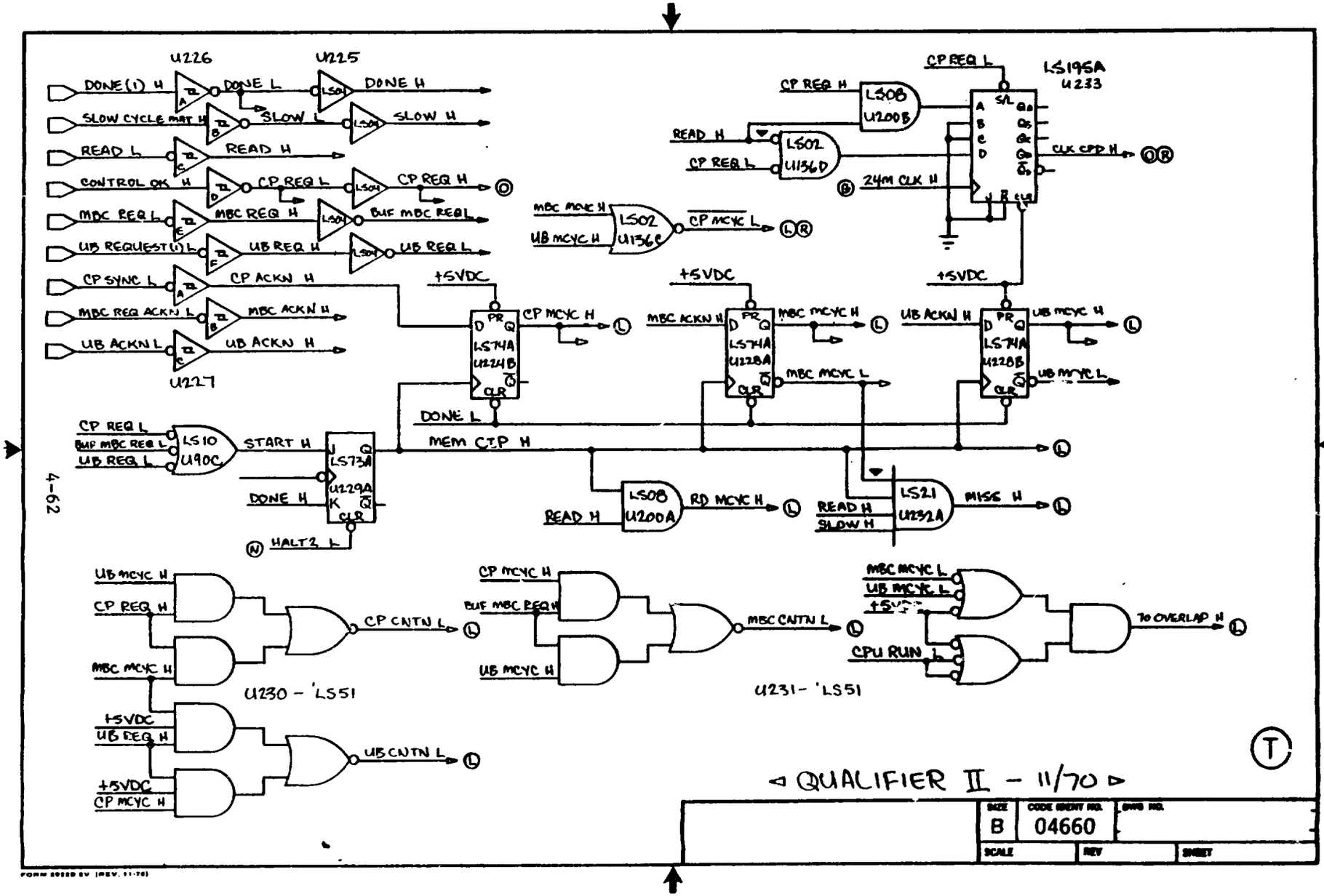
S153
U155
U156
U157
U158
NMC4164-1
S86
U112C
S08
U171A
S74
U172A
S280
U170
LS245
S174
LS10
U24C
ARAM A7 H
GEN A PAR H
APAR BAD H
WR ENB L
AD CAS L
CLR ERR L
ARAM D7 H
ARAM D1 H
AND 15 H
MBD 15 H
AD ROW L
BUF RFSH H
RD DATA L
ENB MC ACC L
ADSTROBE H
HALT L
HALT H
MC REQ H
◁ ADDRESS ASSOCIATIVE MEMORY ▷
B
04660

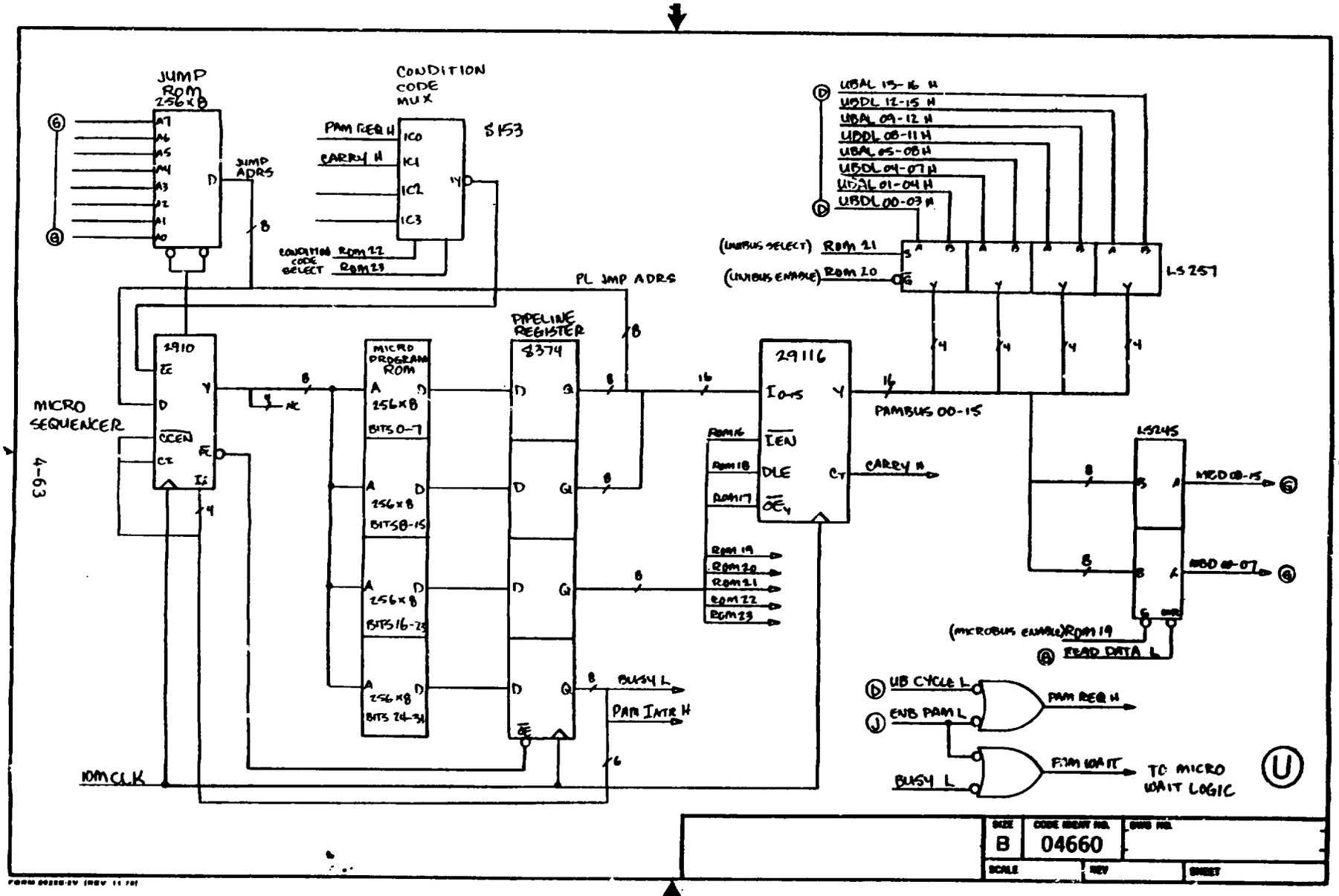

S153
U173
U174
U175
U176
MC A7 H
RFSH A7 H
CPU D15 H
07
MC A6 H
RFSH A6 H
CPU D14 H
06
MC A5 H
RFSH A5 H
CPU D13 H
05
MC A4 H
RFSH A4 H
CPU D12 H
04
MC A3 H
RFSH A3 H
CPU D11 H
03
MC A2 H
RFSH A2 H
CPU D10 H
02
MC A1 H
RFSH A1 H
CPU D09 H
01
MC A0 H
RFSH A0 H
CPU D08 H
00
AD ROW L
DRAM A7 H
NMC4164 -1
U177 U178 U179 U180 U181
WR ENB L
AD RAS L
AD CAS L
RD D PAR H
GEN D PAR H
S86
U112D
WR ENB L
D PAR BAD H
S08
U171B
S74
U172B
AD CAS L
CLR ERR L
D PAR ERR H
D PAR ERR L
S280
U181
PULL H
RQB H
S74
U215A
ZOOM CLK L
AD ROW L
HALT1 L
S174
U183
ARAM D1 H
DRAM D5 H
AM D09 H
LS245
U182
MBD 07 H
CRAM D5 H
RD DATA L
ENB MC ACC L
ADSTROBE H
HALT L
'28
U53D
DATA CYC H
BUF RFSH H
DATA ASSOCIATNE MEMORY
Q
4-58
SIZE
B
CODE IDENT NO.
04660
DWG NO.
SCALE
REV
SHEET


NMC4164-1
S153
U165
U166
U167
U168
MC A7 H
RFSH A7 H
CPU A15 H
07
MC A6 H
RFSH A6 H
CPU A14 H
06
MC A5 H
RFSH A5 H
CPU A13 H
05
MC A4 H
RFSH A4 H
CPU A12 H
04
MC A3 H
RFSH A3 H
CPU A11 H
03
MC A2 H
RFSH A2 H
CPU A10 H
02
MC A1 H
RFSH A1 H
CPU A09 H
01
MC A0 H
RFSH A0 H
CPU A08 H
00
AD ROW L
BUF RFSH H
ARAM A7 H
WR ENB L
AD RAS L
AD CAS L
U160 U161 U162 U163 U164 U165 U166 U167
RD A PAR H
S86
U112C
APAR BAD H
S08
U171A
WR ENB L
S74
U172A
A PAR ERR H
A PAR ERR L
AD CAS L
CLR ERR L
GEN A PAR H
S280
U170
NC
LS245
U168
MBD 15 H
ARAM D7 H
ARAM DI H
S174
U169
AMD 15 H
BUS AK L
HALT H
MC REQ H
LS10
U24C
RD DATA L
END MC ACC L
ADSTROBE H
HALT L
ADDRESS ASSOCIATIVE MEMORY
SIZE
B
CODE IDENT NO.
04660
DWG NO.
SCALE
REV
SHEET
4-59


LS378
U185
U186
KERNEL L
SUPER L
USER L
PRIOR 7 L
PRIOR 6 L
FETCH L
CLK CPD H
MC A7 H
RFSH A7 H
AM D11 H
UBCLOSH H
MC A6 H
RFSH A6 H
AM D10 H
UBCLOBH H
MC A5 H
RFSH A5 H
AM D09 H
UBCLO7H H
MC A4 H
RFSH A4 H
AM D08 H
UBCLO6 H
MC A3 H
RFSH A3 H
AM D05 H
UBCLO5H H
MC A2 H
RFSH A2 H
AM D04 H
UBCLO4 H
MC A1 H
RFSH A1 H
UBCLO3H H
MC A0 H
RFSH A0 H
UBCLO1H H
U187
U188
U189
U190
CRAM A7 H
WR ENB L
CRAS L
CCAS L
NMC4164-1
U191
U192
U193
U194
U195
ECAS L
S74
U196A
CPARERRH
EPARERRL
CLR ERR
S08
U171C
WREBL
C PAR BAD H
RD C PAR H
GEN C PAR H
S86
U197A
S280
U198
CRAM D0 H
S175
U199
AM D05 H
OR
01
00
C STROBE H
HALT L
PULL H
QB H
20MCLK L
U213B
C ROW L
BUF RFSH H
MBWR L
LS32
U60D
WR ENB L
HALT L
COMBINATIONAL ASSOCIATIVE MEMORY
B
04660
SCALE
REV
SHEET
4-60


MASK09 H
Am DI5 H
LS08 U217A
MAM 09 H
MASK08 H
Am DI4 H
LS08 U217B
MAM 08 H
MASK07 H
Am DI3 H
LS08 U217C
MAM 07 H
MASK06 H
Am DI2 H
LS08 U217D
MAM 06 H
MASK05 H
Am DI7 H
LS08 U219A
MAM 05 H
MASK04 H
Am DI6 H
LS08 U219B
MA. 104 H
MASK03 H
Am DO3 H
LS08 U218A
MAM 03 H
MASK02 H
Am DO2 H
LS08 U218B
MAM 02 H
DLY CSTB L
LS73A U220A
GATE 3 H
MASK01 H
Am DO1 H
LS08 U218C
MAM 01 H
MASK00 H
Am DO0 H
LS08 U218D
MAM 00 H
HALT L
C STROBE H
180 ns
LS132 U215B
DLY CSTB L
LS73A U220B
GATE 1 H
AD STROBE H
400 ns
LS132 U215C
DLY ADSTB L
BBSY H
IND RUN H
LS08 U219C
UB OVER LAP H
U138F
LS04
CPU RUN L
LS139 U122
PS 06 H
PS 05 H
MMRD MODE 1 H
MMRD MODE 0 H
PRIOR 7 L
6
5
4
USER L
NC
SUPER L
KERNEL L
PS07 H
LS04
PS07 L
U227
IND PAUSE H
PAUSE L
UIRK H
FETCH L
IND MASTER H
MASTER L
LS14
1 MS
LS132 U215D
DLY REQ L
LS02 U136B
FP WAIT H
PULL H
FP ATTN L
FP11-B
FP START (0) H
FP11-C
U125B
LS04
START FP H
LS74A U223A
FP SYNC L
FP EXC TRAP L
HALT H
INTR H
LS260 U122A
21M CLK H
FP REQ L
QUALIFIER I
SIZE B
CODE IDENT NO. 04660
DWG NO.
SCALE
REV
SHEET


U226
U225
DONE (I) H
DONE L
DONE H
SLOW CYCLE REQ H
SLOW L
SLOW H
READ L
READ H
CONTROL OK H
CP REQ L
CP REQ H
MBC REQ L
MBC REQ H
BUF MBC REQ L
UB REQUEST (I) L
UB REQ H
UB REQ L
CP SYNC L
CP ACKN H
MBC REQ ACKN L
MBC ACKN H
UB ACKN L
UB ACKN H
U227
CP REQ H
LS08
U200B
READ H
LS02
U113G D
CP REQ L
24M CLK H
LS195A
U233
CLK CPP H
MBC MCYC H
UB MCYC H
LS02
U113G C
CP MCYC L
+5VDC
LS74A
U124B
CP MCYC H
U122B
MBC MCYC H
U122A
MBC MCYC L
UB ACKN H
U122B
UB MCYC H
UB MCYC L
DONE L
MEM CYC H
CP REQ L
BUF MBC REQ L
UB REQ L
LS10
U90C
START H
LS73A
U129A
DONE H
HALT2 L
LS08
U200A
RD MCYC H
READ H
SLOW H
LS21
U132A
MISS H
UB MCYC H
CP REQ H
CP CNTN L
MBC MCYC H
+5VDC
UB REQ H
UB CNTN L
+5VDC
CP MCYC H
U230 - 'LS51
CP MCYC H
BUF MBC REQ H
MBC CNTN L
UB MCYC H
U231 - 'LS51
MBC MCYC L
UB MCYC L
+5VDC
CPU RUN L
TO OVERLAP H
QUALIFIER II - 11/70
SIZE
B
CODE IDENT NO.
04660
DWG NO.
SCALE
REV
SHEET
4-62



SIZE	CODE IDENT NO.	DWG NO.
B	04660	
SCALE	REV	SHEET

SECTION 5.0

PRELIMINARY PARTS LIST

(MAJOR COMPONENTS)

and

TIMING DIAGRAMS

	SIZE	CODE IDENT NO.	DWG. NO.
5-1	A	04660	
	SCALE		SHEET

FORM 2022A-IV-1 (REV. 9-73)

PRELIMINARY PARTS LIST (MAJOR COMPONENTS)

ITEM	PART NO.	DESCRIPTION	LOCATION	QUANTITY
1	SN74LS00	Integrated Circuit	U51, U89, U202	3
2	SN74LS02	" "	U10, U109, U136	3
3	SN74LS04	"	U11, U71, U138, U225	4
4	SN74LS08	" "	U12, U25, U59, U91, U135, U200, U217-U219	9
5	SN74S08	" "	U171	1
6	SN74LS10	" "	U24, U90	2
7	SN74LS11	" "	U141	1
8	SN74LS14	" "	U226, U227	2
9	SN74LS20	" "	U13	1
10	SN74LS21	" "	U203. U232	2
11	SN74LS27	" "	U73, U154	2
12	SN7428	" "	U33	1
13	SN74LS30	" "	U206, U214	2
14	SN74LS32	" "	U9, U55, U60, U75, U211	5
15	SN74LS51	" "	U230, U231	2
16	SN74LS73A	" "	U220, U229	2
17	SN74LS74A	" "	U14, U19, U21, U23, U26, U42, U52, U54, U56, U58 U74, U83, U117, U137, U207, U210, U212, U223, U224, U228	20
18	SN74S74	" "	U131, U172, U196, U213	4
19	SN74S86	" "	U112, U197	2
20	SN74S112	" "	U133	1
21	SN74LS114A	" "	U57, U201	2
22	SN74S114	" "	U132, U134	2
23	SN74LS123	" "	U53	1
24	SN74LS132	" "	U27, U215	2
25	SN74LS133	" "	U205	1
26	SN74LS138	" "	U63, U72, U113-U115, U221	6
27	SN74LS139	" "	U8	1
28	SN74S153	" "	U155-U158, U173-U176, U187-U190	12
29	SN74LS157	" "	U142-U145	4

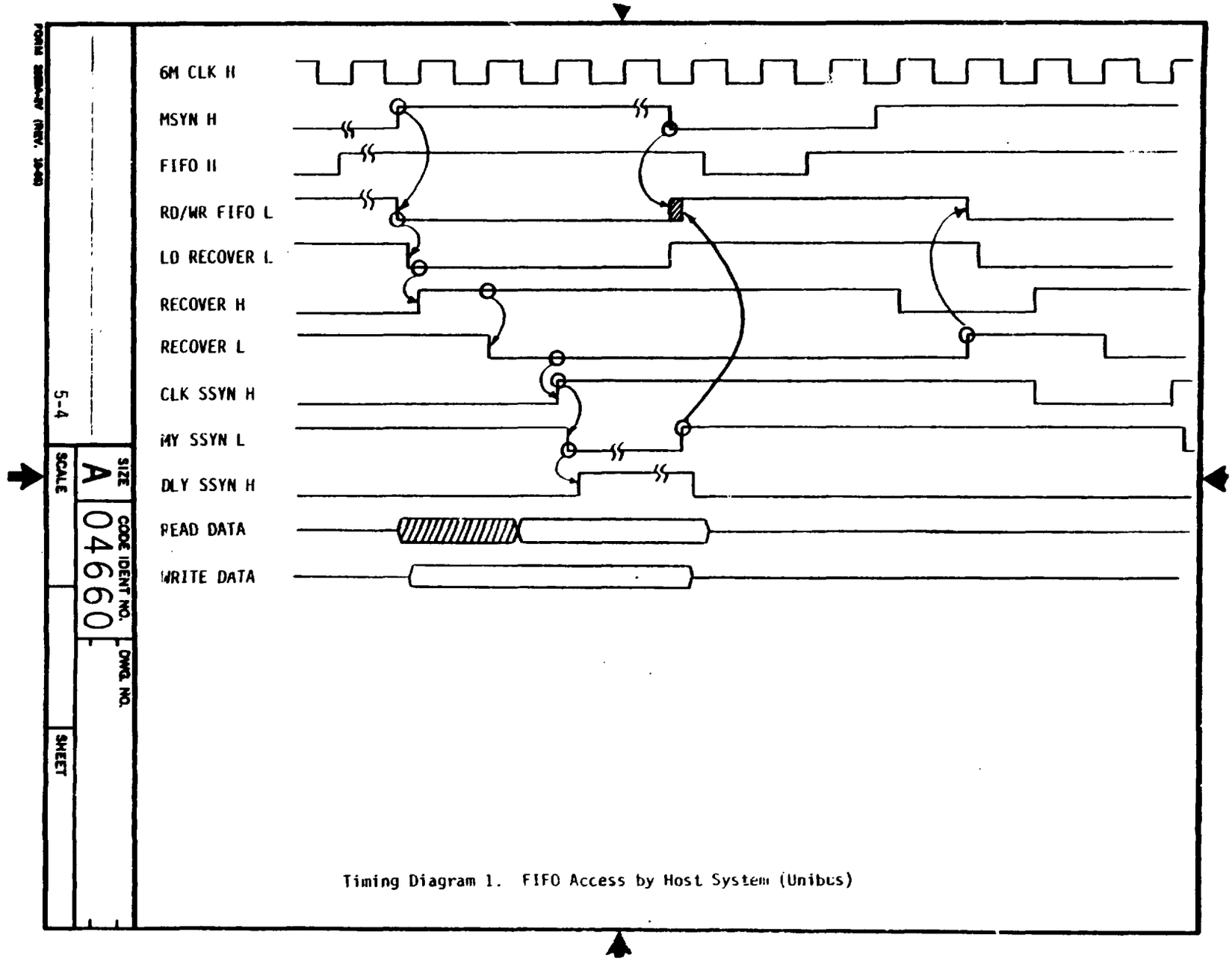
SIZE	CODE IDENT NO.	DWG. NO.
A	04660	
SCALE		SHEET

	ITEM	PART NO.	DESCRIPTION	LOCATION	QUANTITY
	29-a	SN74S157	Integrated Circuit	U86, U87, U152, U153	4
	30	SN74S158	" "	U139	1
	31	SN74LS161A	" "	U140, U150, U151	3
	32	SN74S174	" "	U169, U183	2
	33	SN74S175	" "	U199	1
	34	SN74LS195A	" "	U22, U76, U116, U204	4
	35	SN74S225	" "	U49, U50	2
	36	SN74LS244	" "	U43-U46, U61, U208, U209	7
	37	SN74LS245	" "	U69, U70, U168, U182	4
	38	SN74LS260	" "	U77, U222	2
	39	SN74S280	" "	U110, U111, U170, U184, U198	5
	40	SN74S244	" "	U65	1
	41	SN74LS373	" "	U66-U68	3
	42	SN74LS374	" "	U38-U41, U146-U149	8
	43	SN74LS378	" "	U185, U186	2
	44	DS8641	" "	U1-U5, U15-U18, U28-U32	14
	45	AmZ8001A	" "	U64	1
	46	AmZ8127	" "	U62	1
	47	Z8038A	" "	U47, U48	2
	48	Am9513	" "	U118-U125	8
	49	Am9519-1	" "	U126-U130	5
	50	Am25LS2518	" "	U20, U34-U37	5
	51	Am25LS2521	" "	U6, U7	2
	52	HM-7616	" "	U78-U85	2-8
*	52	2716	" "	U78-U85	2-8
*	52	2732	" "	U78-U85	2-8
	53	NMC5295	" "	U91-U108	18
**	53	NMC4164	" "	U91-U108	18
**	53	2118	" "	U91-U108	18
	54	NMC4164-1	" "	U159-U167, U177-U181, U191-U195	19
	55	CO-238A	Oscillator	40 MHZ	1

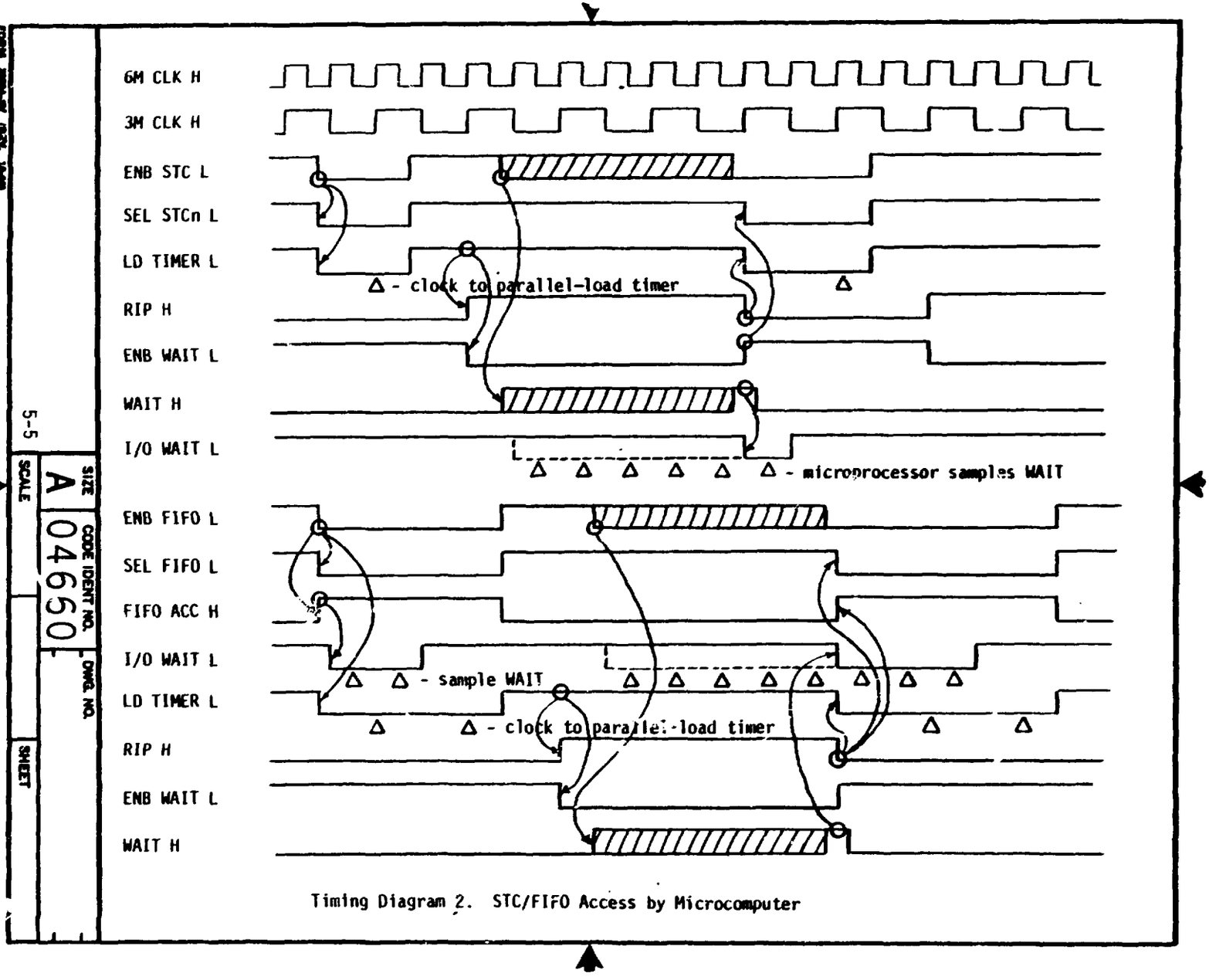
*, ** represent alternate parts

SIZE	CODE IDENT NO.	DWG. NO.
A	04660	

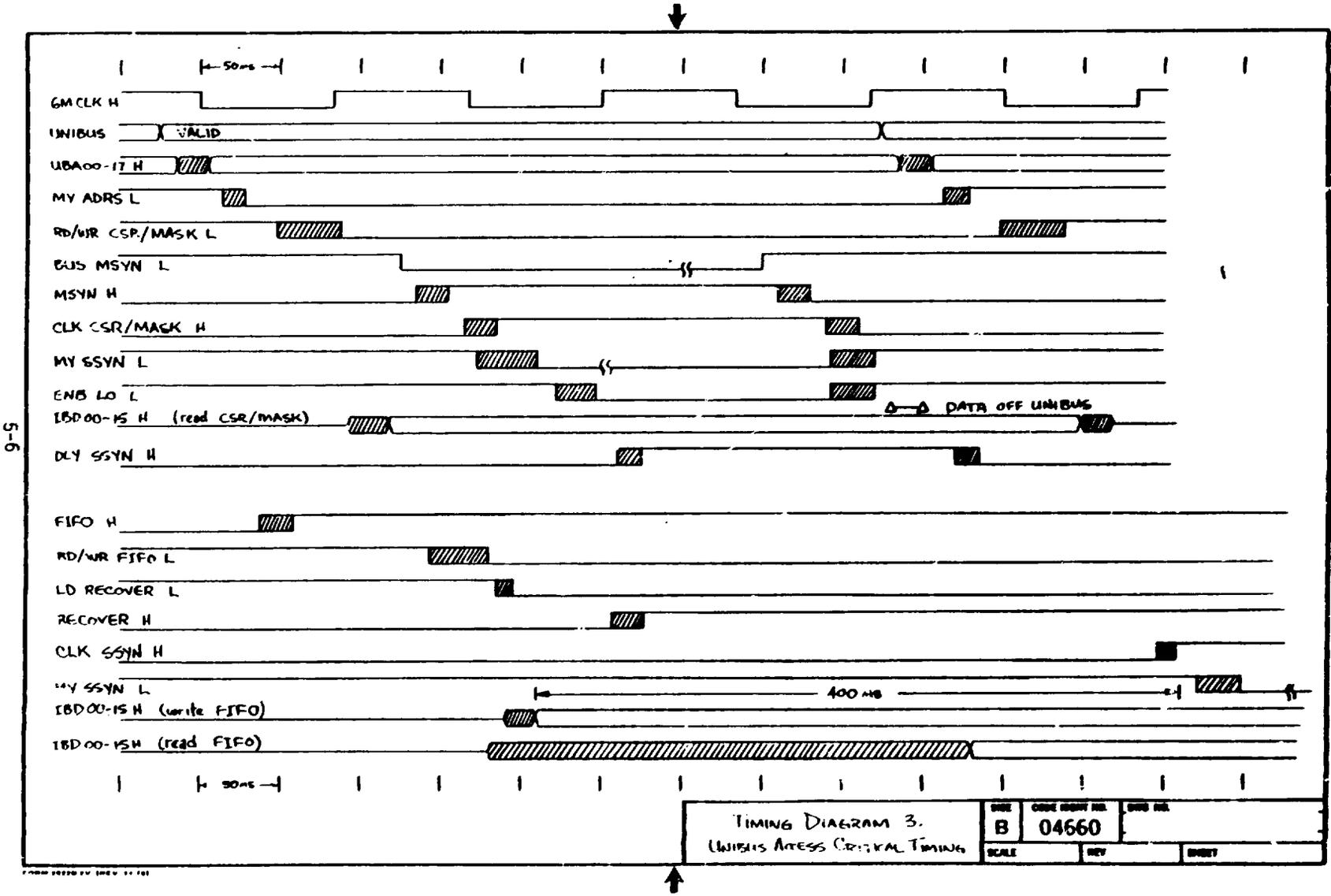
SCALE		SHEET

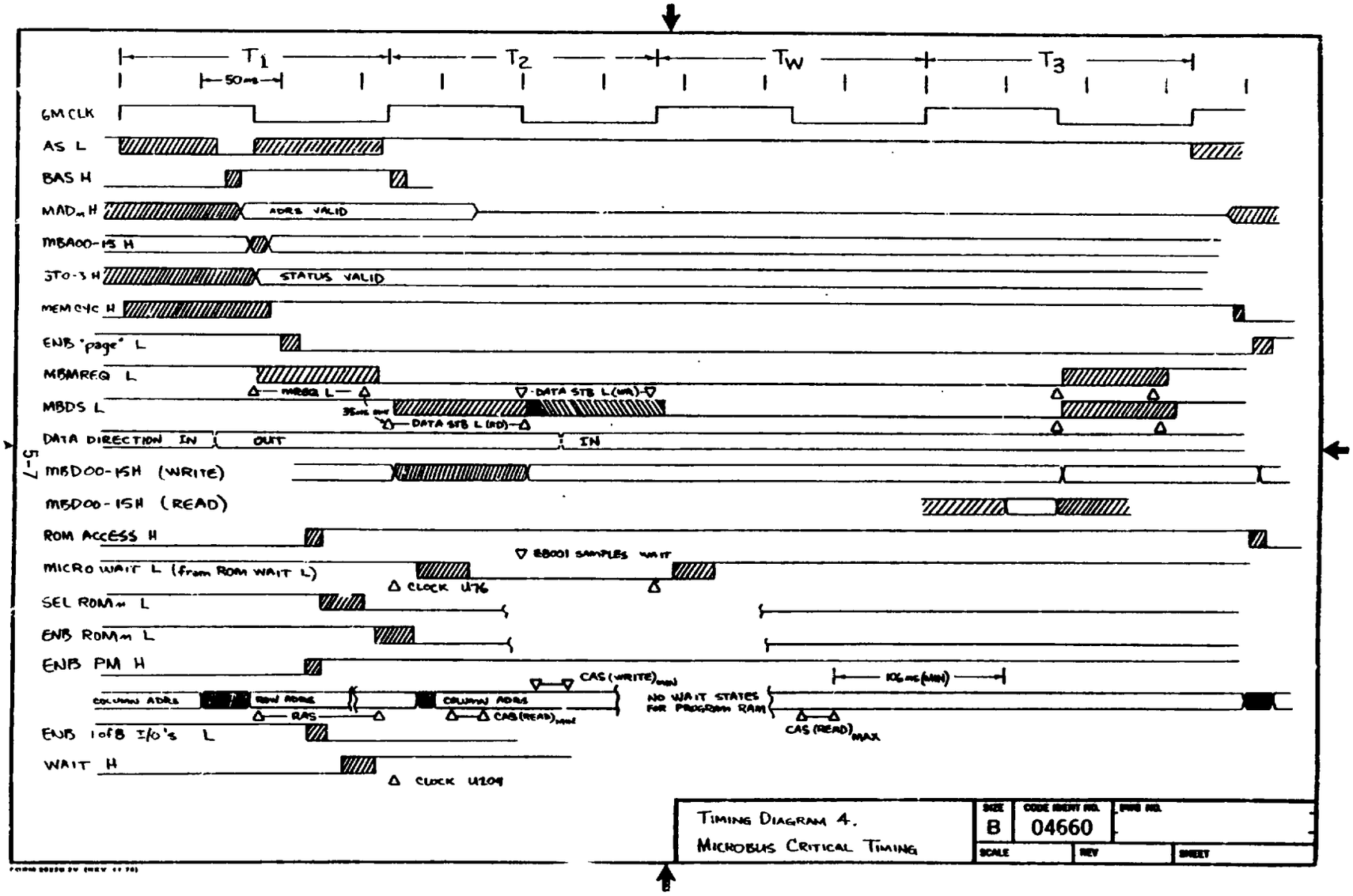


Timing Diagram 1. FIFO Access by Host System (Unibus)



Timing Diagram 2. STC/FIFO Access by Microcomputer


50ns
GM CLK H
UNIBUS
VALID
UBA00-17 H
MY ADRS L
RD/WR CSR/MASK L
BUS MSYN L
MSYN H
CLK CSR/MASK H
MY SSYN L
ENB LO L
IBD00-15 H (read CSR/MASK)
DATA OFF UNIBUS
DLY SSYN H
FIFO H
RD/WR FIFO L
LD RECOVER L
RECOVER H
CLK SSYN H
MY SSYN L
400 ns
IBD00-15 H (write FIFO)
IBD00-15 H (read FIFO)
50ns
Timing Diagram 3.
UNIBUS Access Critical Timing
B
04660

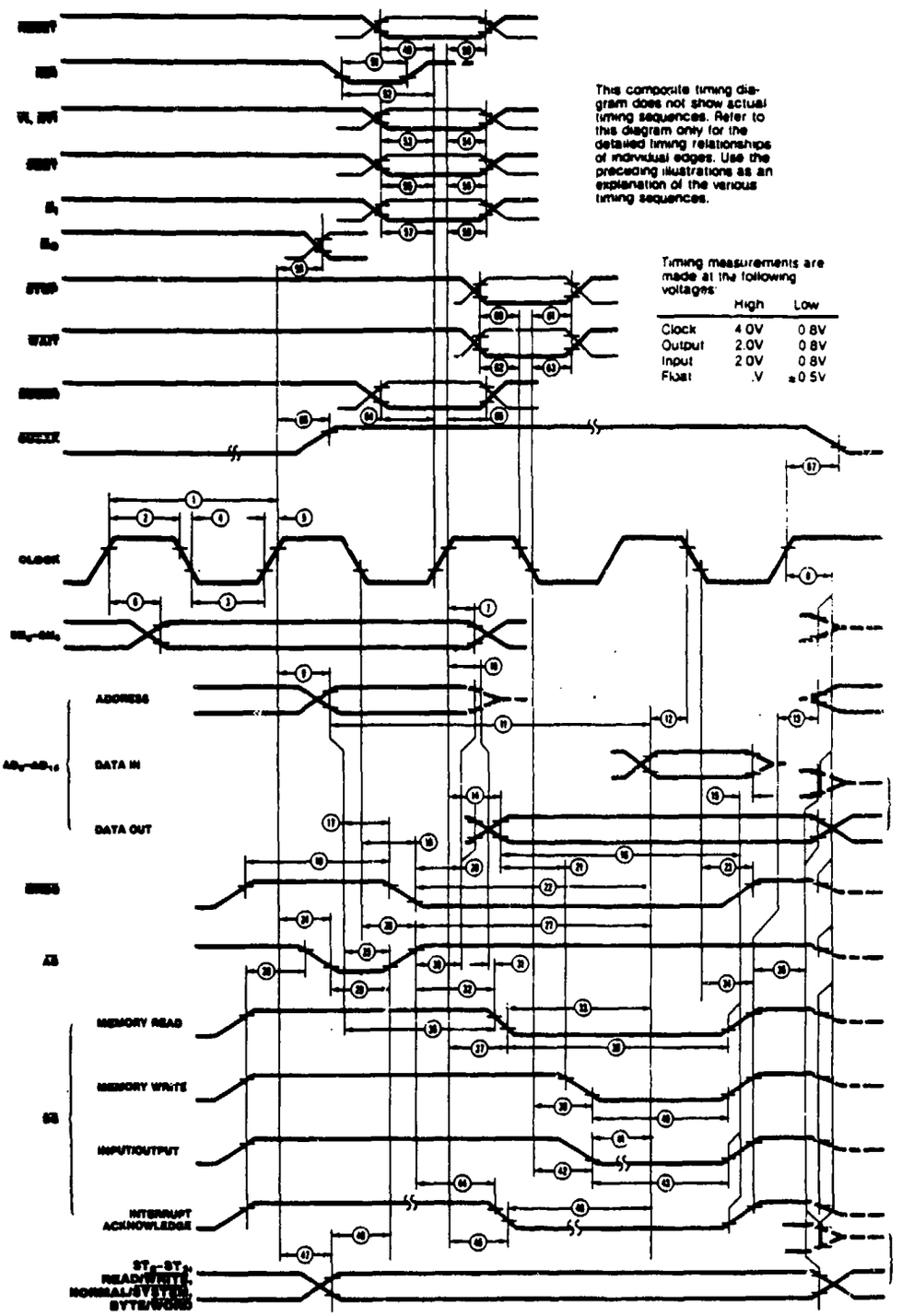

T1
T2
TW
T3
50ns
GM CLK
AS L
BAS H
MADm H
ADRS VALID
MBA00-15 H
STO-3 H
STATUS VALID
MEM CYC H
ENB "page" L
MBMREQ L
MREQ L
DATA STB L (WR)
MBDS L
35ns min
DATA STB L (RD)
DATA DIRECTION IN
OUT
IN
MBD00-15H (WRITE)
MBD00-15H (READ)
ROM ACCESS H
8B001 SAMPLES WAIT
MICRO WAIT L (from ROM WAIT L)
CLOCK U76
SEL ROMm L
ENB ROMm L
ENB PM H
CAS (WRITE)min
106ns (MIN)
COLUMN ADRS
ROW ADRS
COLUMN ADRS
NO WAIT STATES FOR PROGRAM RAM
RAS
CAS (READ)min
CAS (READ)MAX
ENB 1 of 8 I/O's L
WAIT H
CLOCK U109
TIMING DIAGRAM 4.
MICROBUS CRITICAL TIMING
SIZE
B
CODE IDENT NO.
04660
DWG NO.
SCALE
REV
SHEET

AC Characteristics

Number	Symbol	Parameter	Z8001/Z8002 Min (ns)	Z8001/Z8002 Max (ns)	Z8001A/Z8002A Min (ns)	Z8001A/Z8002A Max (ns)
1	TcC	Clock Cycle Time	250	2000	165	2000
2	TwCh	Clock Width (High)	105	2000	70	2000
3	TwCl	Clock Width (Low)	105	2000	70	2000
4	TfC	Clock Fall Time		20		10
5	TrC	Clock Rise Time		20		10
6	TdC(SNv)	Clock ↑ to Segment Number Valid (50 pF load)		130		110
7	TdC(SNn)	Clock ↑ to Segment Number Not Valid	20		10	
8	TdC(Bz)	Clock ↑ to Bus Float		65		55
9	TdC(A)	Clock ↑ to Address Valid		100		75
10	TdC(Az)	Clock ↑ to Address Float		65		55
11	TdA(DI)	Address Valid to Data In Required Valid	455*		305*	
12	TsDI(C)	Data In to Clock ↓ Setup Time	50		20	
13	TdDS(A)	$\overline{DS}$ ↑ to Address Active	80*		40*	
14	TdC(DO)	Clock ↑ to Data Out Valid		100		75
15	ThDI(DS)	Data In to $\overline{DS}$ ↑ Hold Time	0		0	
16	TdDO(DS)	Data Out Valid to $\overline{DS}$ ↑ Delay	295*		195*	
17	TdA(MR)	Address Valid to MREQ ↓ Delay	(55)*		(35)*	
18	TdC(MR)	Clock ↓ to MREQ ↓ Delay		80		70
19	TwMRh	MREQ Width (High)	210*		135*	
20	TdMR(A)	MREQ ↑ to Address Not Active	70*		35*	
21	TdDO(DSW)	Data Out Valid to $\overline{DS}$ ↓ (Write) Delay	55*		35*	
22	TdMR(DI)	MREQ ↑ to Data In Required Valid	350*		225*	
23	TdC(MR)	Clock ↓ MREQ ↑ Delay		80		60
24	TdC(ASf)	Clock ↓ to $\overline{AS}$ ↓ Delay		80		60
25	TdA(AS)	Address Valid to $\overline{AS}$ ↑ Delay	55*		35*	
26	TdC(ASr)	Clock ↓ to $\overline{AS}$ ↑ Delay		90		80
27	TdAS(DI)	$\overline{AS}$ ↑ to Data In Required Valid	340*		215*	
28	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	70*		35*	
29	TwAS	$\overline{AS}$ Width (Low)	85*		55*	
30	TdAS(A)	$\overline{AS}$ ↑ to Address Not Active Delay	60*		30*	
31	TdAz(DSR)	Address Float to $\overline{DS}$ (Read) ↓ Delay	0		0	
32	TdAS(DSR)	$\overline{AS}$ ↓ to $\overline{DS}$ (Read) ↓ Delay	70*		35*	
33	TdDSR(DI)	$\overline{DS}$ (Read) ↓ to Data In Required Valid	185*		130*	
34	TdC(DSr)	Clock ↓ to $\overline{DS}$ ↑ Delay		70		65
35	TdDS(DO)	$\overline{DS}$ ↑ to Data Out and STATUS Not Valid	75*		45*	
36	TdA(DSR)	Address Valid to $\overline{DS}$ (Read) ↓ Delay	180*		110*	
37	TdC(DSR)	Clock ↑ to $\overline{DS}$ (Read) ↓ Delay		120		85
38	TwDSR	$\overline{DS}$ (Read) Width (Low)	275*		185*	
39	TdC(DSW)	Clock ↑ to $\overline{DS}$ (Write) ↓ Delay		95		80
40	TwDSW	$\overline{DS}$ (Write) Width (Low)	185*		110*	
41	TdDSI(DI)	$\overline{DS}$ (Input) ↓ to Data In Required Valid	320*		200*	
42	TdC(DSf)	Clock ↓ to $\overline{DS}$ (I/O) ↓ Delay		120		100
43	TwDS	$\overline{DS}$ (I/O) Width (Low)	410*		255*	
44	TdAS(DSA)	$\overline{AS}$ ↑ to $\overline{DS}$ (Acknowledge) ↓ Delay	1065*		690*	
45	TdC(DSA)	Clock ↑ to $\overline{DS}$ (Acknowledge) ↓ Delay		120		85
46	TdDSA(DI)	$\overline{DS}$ (Ack.) ↓ to Data In Required Delay	435*		295*	
47	TdC(S)	Clock ↑ to Status Valid Delay		110		85
48	TdS(AS)	Status Valid to $\overline{AS}$ ↑ Delay	50*		30*	
49	TsR(C)	$\overline{RESET}$ to Clock ↑ Setup Time	180		70	
50	ThR(C)	$\overline{RESET}$ to Clock ↑ Hold Time	0		0	
51	TwNMI	$\overline{NMI}$ Width (Low)	100		70	
52	TsNMI(C)	$\overline{NMI}$ to Clock ↑ Setup Time	140		70	
53	TsVI(C)	$\overline{VI}$, NVI to Clock ↑ Setup Time	110		50	
54	ThVI(C)	$\overline{VI}$, NVI to Clock ↑ Hold Time	0		0	
55	TsSGT(C)	$\overline{SEGT}$ to Clock ↑ Setup Time	70		55	
56	ThSGT(C)	$\overline{SEGT}$ to Clock ↑ Hold Time	0		0	
57	TsMI(C)	$\overline{MI}$ to Clock ↑ Setup Time	180		110	
58	ThMI(C)	$\overline{MI}$ to Clock ↑ Hold Time	0		0	
59	TdC(MO)	Clock ↑ to MO Delay		120		85
60	TsSTP(C)	$\overline{STOP}$ to Clock ↑ Setup Time	140		70	
61	ThSTP(C)	$\overline{STOP}$ to Clock ↑ Hold Time	0		0	
62	TsWT(C)	$\overline{WAIT}$ to Clock ↑ Setup Time	50		30	
63	ThWT(C)	$\overline{WAIT}$ to Clock ↑ Hold Time	10		10	
64	TsBRQ(C)	$\overline{BUSRQ}$ to Clock ↑ Setup Time	90		80	
65	ThBRQ(C)	$\overline{BUSRQ}$ to Clock ↑ Hold Time	10		10	
66	TdC(BAKr)	Clock ↑ to $\overline{BUSAK}$ ↑ Delay		100		75
67	TdC(BAKf)	Clock ↑ to $\overline{BUSAK}$ ↓ Delay		100		75

*Clock-cycle-time-dependent characteristics. See table on back page.

Composite AC Timing Diagram



This composite timing diagram does not show actual timing sequences. Refer to this diagram only for the detailed timing relationships of individual edges. Use the preceding illustrations as an explanation of the various timing sequences.

Timing measurements are made at the following voltages:

	High	Low
Clock	4.0V	0.8V
Output	2.0V	0.8V
Input	2.0V	0.8V
Float	.V	±0.5V

Z8000 (3)

Clock-Cycle-Time-Dependent Characteristics	NUMBER	SYMBOL	Z8001/Z8002 EQUATION	Z8001A/Z8002A EQUATION
	11	TdA(DI)	2TcC + TwCh - 150 ns	2TcC + TwCh - 95 ns
	13	TdDS(C)	TwCl - 25 ns	TWCl - 30 ns
	16	TdDO(DS)	TcC + TwCh - 60 ns	TcC + TwCh - 40 ns
	17	TdA(MR)	TwCh - 50 ns	TwCh - 35 ns
	19	TwMRh	TcC - 40 ns	TcC - 30 ns
	20	TdMR(A)	TwCl - 35 ns	TwCl - 35 ns
	21	TdDO(DSW)	TwCh - 50 ns	TwCh - 35 ns
	22	TdMR(DI)	2TcC - 150 ns	2TcC - 105 ns
	25	TdA(AS)	TwCh - 50 ns	TwCh - 35 ns
	27	TdAS(DI)	2TcC - 160 ns	2TcC - 115 ns
	28	TdDS(AS)	TwCl - 35 ns	TwCl - 35 ns
	29	TwAS	TwCh - 20 ns	TwCh - 15 ns
	30	TdAS(A)	TwCl - 45 ns	TwCl - 40 ns
	32	TdAS(DSR)	TwCl - 35 ns	TwCl - 35 ns
	33	TdDSR(DI)	TcC + TwCh - 170 ns	TcC + TwCh - 105 ns
	35	TdDS(DO)	TwCl - 30 ns	TwCl - 25 ns
	36	TdA(DSR)	TcC - 70 ns	TcC - 55 ns
	38	TwDSR	TcC + TwCh - 80 ns	TcC + TwCh - 50 ns
	40	TwDSW	TcC - 65 ns	TcC - 55 ns
	41	TdDSI(DI)	2TcC - 180 ns	2TcC - 130 ns
	43	TwDS	2TcC - 90 ns	2TcC - 75 ns
	44	TdAS(DSA)	4TcC + TwCl - 40 ns	4TcC + TwCl - 40 ns
	46	TdDSA(DI)	2TcC + TwCh - 170 ns	2TcC + TwCh - 105 ns
	48	TdS(AS)	TwCh - 55 ns	TwCh - 40 ns

Z8038 (1)

Z-Bus CPU Interface Timing	Number	Symbol	Parameter	Min	Max	Units	Notes
	1	TwAS	AS Low Width	70		ns	
	2	TsA(AS)	Address to AS ↓ Setup Time	10		ns	1
	3	ThA(AS)	Address to AS ↓	50		ns	1
	4	TsCSO(AS)	CS to AS ↓ Setup	0		ns	1
	5	ThCSO(AS)	CS to AS ↓ Hold Time	60		ns	1
	6	TdAS(DS)	AS ↑ to DS ↓ Delay	60		ns	1
	7	TsA(DS)	Address to DS ↓	120		ns	
	8	TsRWR(DS)	R/W (Read) to DS ↓ Setup Time	100		ns	
	9	TsRWW(DS)	R/W (Write) to DS ↓ Setup Time	0			
	10	TwDS	DS Low Width	380		ns	
	11	TsDW(DSf)	Write Data to DS ↓ Setup Time	30		ns	
	12	TdDS(DRV)	DS (Read) ↓ to Address Data Bus Driven	0			
	13	TdDS(DR)	DS ↓ to Read Data Valid Delay		255	ns	
	14	ThDW(DS)	Write Data to DS ↑ Hold Time	30		ns	
	15	TdDSr(DR)	DS ↑ to Read Data Not Valid Delay	0			
	16	TdDS(DRz)	DS ↑ to Read Data Float Delay		70	ns	2
	17	ThRW(DS)	R/W to DS ↑ Hold Time	60		ns	
	18	TdDS(AS)	DS ↑ to AS ↓ Delay	50		ns	
	19	Trc	Valid Access Recovery Time	1000		ns	3

1. Parameter does not apply to Interrupt Acknowledge transactions.
2. Float Delay is measured to the time the output has changed 0.5V from steady state with minimum AC load and maximum DC load.
3. This is the delay from DS of one PIO access to DS of another PIO access (either read or write).

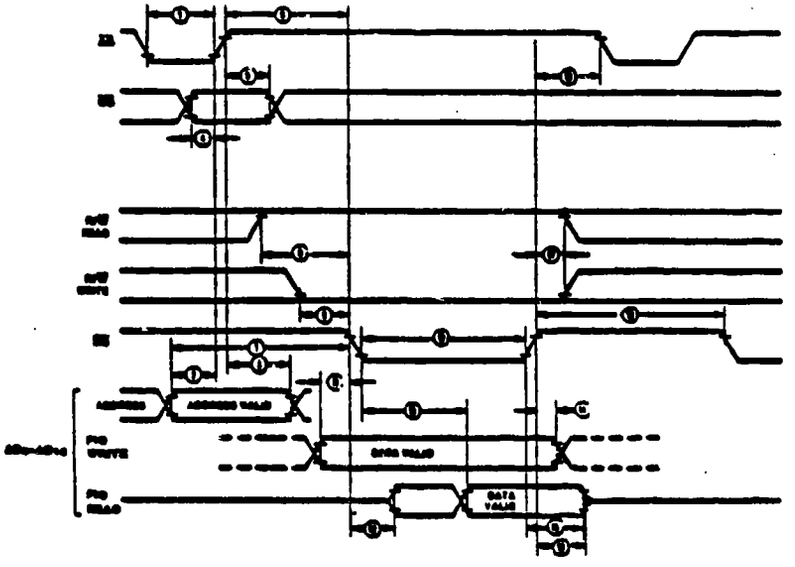


Figure 37. Z-Bus CPU Interface Timing

Z-Bus CPU Interrupt Acknowledge Timing

Number	Symbol	Parameter	Min	Max	Units	Notes
20	TsIA(AS)	INTACK to $\overline{AS}$ ↑ Setup Time	0			
21	ThIA(AS)	INTACK to $\overline{AS}$ ↑ Hold Time	250		ns	
22	TdDSA(DR)	$\overline{DS}$ (Acknowledge) ↓ to Read Data Valid Delay		360	ns	
23	TwDSA	$\overline{DS}$ (Acknowledge) Low Width	475		ns	
24	TdAS(IEO)	$\overline{AS}$ ↓ to IEO ↓ Delay (INTACK Cycle)		380	ns	4
25	TdIEI(IEO)	IEI to IEO Delay		150	ns	4
26	TsIEI(DSA)	IEI to $\overline{DS}$ (Acknowledge) ↓ Setup Time	100		ns	
27	ThIEI(DSA)	IEI to $\overline{DS}$ (Acknowledge) ↓ Hold Time	200		ns	4
28	TdDS(INT)	$\overline{DS}$ (INTACK cycle) to $\overline{INT}$ Delay			ns	
29	TdDCST	Interrupt Daisy Chain Settle Time				4

4. The parameters for the devices in any particular daisy chain must meet the following requirements: The delay from $\overline{AS}$ to $\overline{DS}$ must be greater than the sum of TdASIEO for the highest priority peripheral, TsIEIDSA) for the lowest priority peripheral, and TdIEI::IEO: for each peripheral connecting them in the chain.

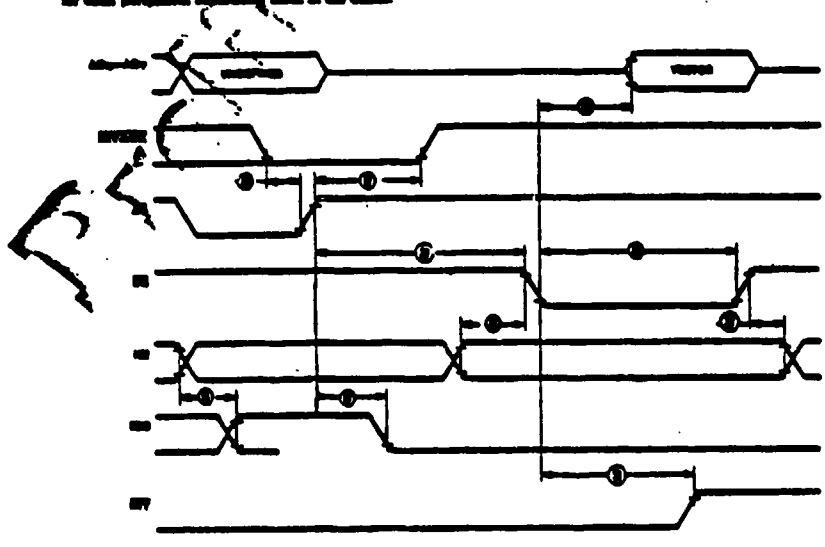


Figure 16. Z-Bus Interrupt Acknowledge Timing

Z-Bus Interrupt Timing	Number	Symbol	Parameter	Min	Max	Units	Notes
	30	TdIM(W(INT)	Message Write to INT Delay		1	AS Cycles + tDS	5
	31	TdIDC(INT)	Data Direction Change to INT Delay		1	AS Cycles + tDS	6
	32	TdIPM(W(INT)	Pattern Match to INT Delay (Write Case)		1	AS Cycles + tDS	
	33	TdIPM(R(INT)	Pattern Match (Read Case) to INT Delay		1	AS Cycles + tDS	
	34	TdISC(INT)	Status Compare to INT Delay		1	AS Cycles + tDS	6
	35	TdIER(INT)	Error to INT Delay		1	AS Cycles + tDS	
	36	TdIEM(INT)	Empty to INT Delay		1	AS Cycles + tDS	6
	37	TdIFL(INT)	Full to INT Delay		1	AS Cycles + tDS	6
	38	TdAS(INT)	AS to INT Delay			tDS	

5. Write to from other side of FIO.
6. Write can be to either state, depending on programming of FIO.

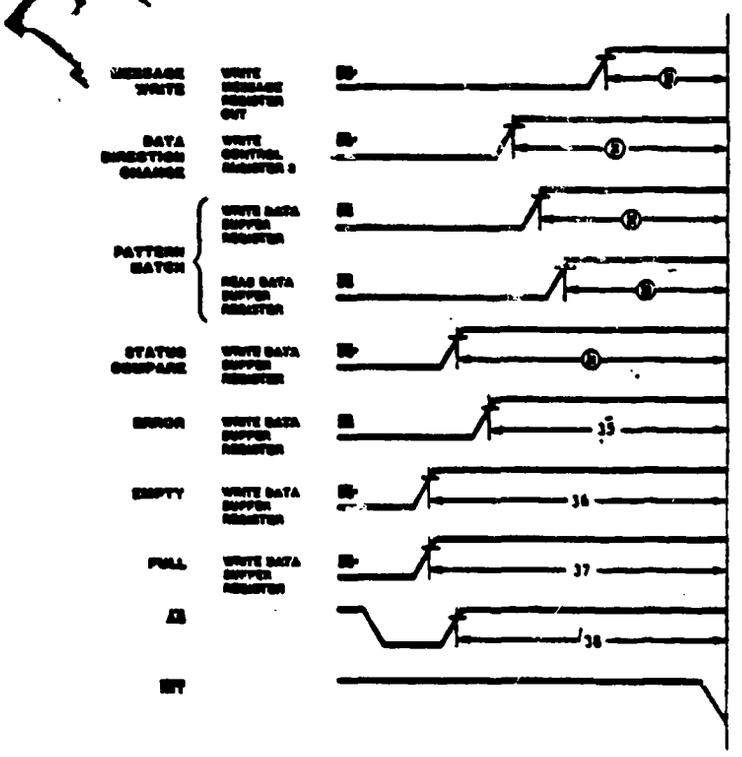


Figure 10. Z-Bus Interrupt Timing

Z-Bus Request Wait Timing

Number	Symbol	Parameter	Min	Max	Units	Notes
1	TdDS(WAIT)	DS ↓ to WAIT ↓ Delay			ns	
2	TdDS↑(WAIT↓)	DS↑ to WAIT ↓ Delay			ns	
3	TdACK(WAIT↓)	ACKIN ↓ to WAIT ↓ Delay			ns	7
4	TdDS(REQ)	DS ↓ to REQ ↑ Delay			ns	
5	TdDMA(REQ)	DMASTB ↓ to REQ ↑ Delay			ns	
6	TdDS↑(REQ↓)	DS↑ to REQ ↓ Delay			ns	
7	TdACK(REQ↓)	ACKIN ↑ to REQ ↓ Delay			ns	
8	TdDMA(DR)	Data Setup Time to DMASTB	200		ns	
9	TdH(DMA)	Data Hold Time to DMASTB	30		ns	
10	TdDMA(DR)	DMASTB ↓ to Valid Data			ns	
11	TdDMA(DRH)	DMASTB ↑ to Data Not Valid	0		ns	
12	TdDMA(DRZ)	DMASTB ↑ to Data Bus Float		70	ns	

9. The delay is from DAV ↓ to 3-State Input Resistor. The delay is from DAV ↑ to 3-State Output Resistance.

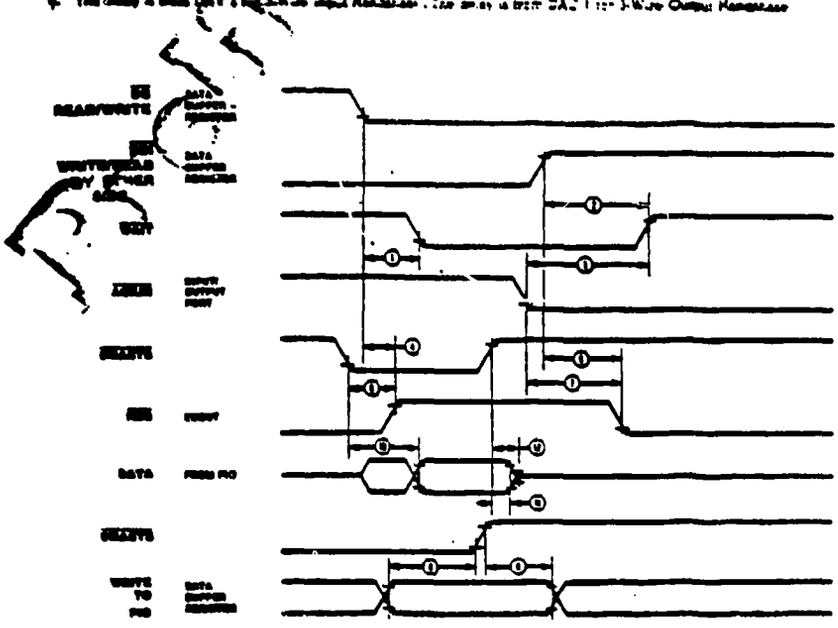


Figure 40. Z-Bus R/W Timing

Z-Bus Reset Timing

Number	Symbol	Parameter	Min	Max	Units	Notes
1	TdDSQ(AS)	Delay from DS ↓ to AS ↓ for no Reset	40		ns	
2	TdASQ(DS)	Delay for AS ↓ to DS ↓ for No Reset	50		ns	
3	Tw(AS • DS)	Minimum width of AS and DS both low for Reset	500		ns	1

1. Internal circuitry allows for the reset provided by the Z-8 (DS held Low while AS pulsed) to be authentic.

Figure 41. Z-Bus RESET Timing

Non Z-Bus CPU Interface Timing

Number	Symbol	Parameter	Min	Max	Units	Notes
1	TsA(RD)	Address Setup to $\overline{RD}$ ↓	80		ns	1
2	TsA(WR)	Address Setup to $\overline{WR}$ ↓	80		ns	
3	ThA(RD)	Address Hold Time to $\overline{RD}$ ↑	0		ns	1
4	ThA(WR)	Address Hold Time to $\overline{WR}$ ↑	0		ns	
5	TsCE(RD)	$\overline{CE}$ Low Setup Time to $\overline{RD}$ ↓	0		ns	1
6	TsCE(WR)	$\overline{CE}$ Low Setup Time to $\overline{WR}$ ↓	0		ns	
7	ThCE(RD)	$\overline{CE}$ Low Hold Time to $\overline{RD}$ ↑	0		ns	1
8	ThCE(WR)	$\overline{CE}$ Low Hold Time to $\overline{WR}$ ↑	0		ns	
9	TsCEh(RD)	$\overline{CE}$ High Setup Time to $\overline{RD}$ ↓	100		ns	1
10	TsCEh(WR)	$\overline{CE}$ High Setup Time to $\overline{WR}$ ↓	100		ns	
11	TwRD1	$\overline{RD}$ Low Width	400		ns	
12	TdRD(DRA)	$\overline{RD}$ ↓ to Read Data Active Delay	0		ns	
13	TdRD(DR)	$\overline{RD}$ ↓ to Valid Data Delay		300	ns	
14	TdRDr(DR)	$\overline{RD}$ ↑ to Read Data Not Valid Delay	0		ns	
15	TdRD(DRz)	$\overline{RD}$ ↑ to Data Bus Float		70	ns	3
16	TwWR1	$\overline{WR}$ low Width	400		ns	
17	TsDW(WR)	Data Setup Time to $\overline{WR}$	0		ns	
18	ThDW(WR)	Data Hold Time to $\overline{WR}$	0		ns	
19	Trv	Valid Access Recovery Time	1000		ns	2

1. Parameter does not apply to Interrupt Acknowledge transactions.
2. This is the delay from $\overline{RD}$ ↑ or $\overline{WR}$ ↑ of one PIO access to $\overline{RD}$ ↓ or $\overline{WR}$ ↓ of another PIO access.
3. Float Delay is measured to the time the output has changed 0.5V from steady state with minimum AC load and maximum DC load.

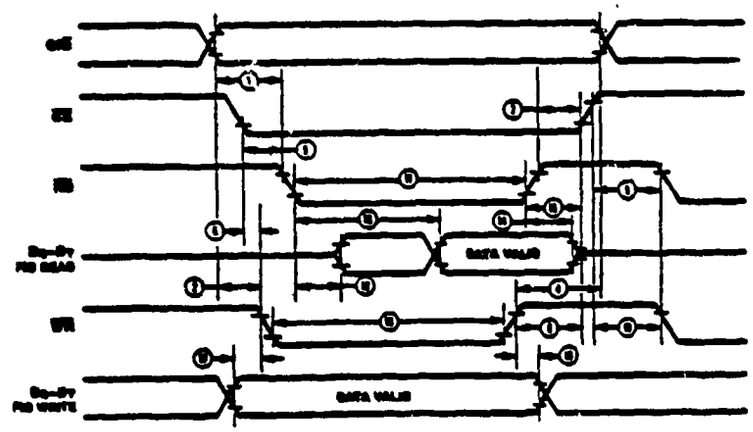


Figure 42. Non-Z-Bus CPU Interface Timing

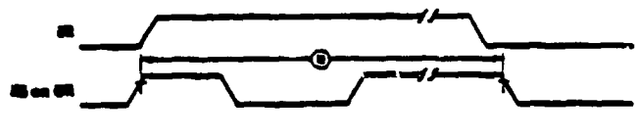


Figure 43. Non-Z-Bus Interface

...-Bus
...errupt
...knowledge
...ing

Number	Symbol	Parameter	Min	Max	Units	Notes
20	TsIEI(IEO)	IEI to IEO Delay	150		ns	4
21	TsIE(IEO)	INTACK↓ to IEO ↓ Delay	300		ns	4
22	TsIEI(RDA)	IEI Setup Time to RD↓ (Acknowledge)	200		ns	4
23	TdRD(DR)	RD↓ to Vector Valid Delay		300	ns	
24	TwRD1(IA)	Read low Width (Interrupt Acknowledge)	400		ns	
25	TsIA(RD)	INTACK↓ to RD↓ Fall Time	30		ns	
26	ThIEI(RD)	IEI Hold Time to RD↑	100		ns	
27	TdRD(INT)	RD↑ to INT↑ Delay			ns	
28	TdDCST	Interrupt Daisy Chain Settle Time				4

4. The parameter for the devices in any particular daisy chain must meet the following statement: The delay from INTACK ↓ to IEI ↓ must be greater than the sum of TdIEI(IEO) for the preceding priority peripheral, TsIEI(RDA) for the in-turn peripheral, and TsDCST(DCST) for each peripheral separating them in the chain.

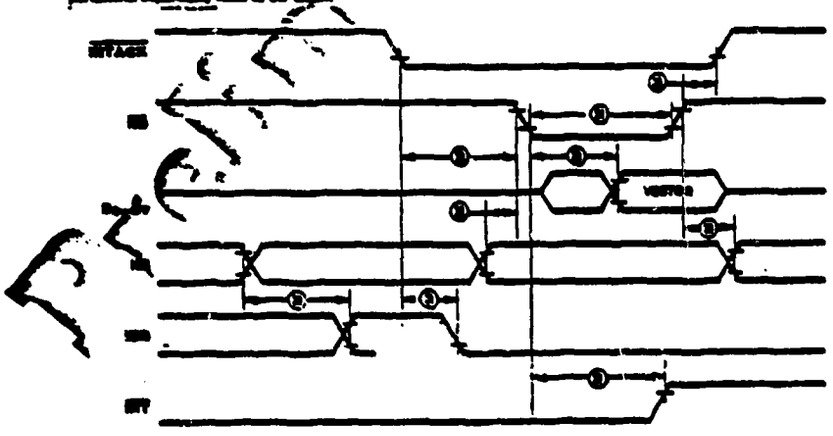


Figure 44. Non-Z-Bus Interrupt Acknowledge Timing

Z8038 (7)

Non-Z-Bus Interrupt Timing

Number	Symbol	Parameter	Min	Max	Units	Notes
29	TdMW(INT)	Message Write to INT Delay			ns	5,6
30	TdDC(INT)	Data Direction Change to INT Delay			ns	5,7
31	TdPMW(INT)	Pattern Match (Write Case) to INT Delay			ns	5
32	TdPMR(INT)	Pattern Match (Read Case) to INT Delay			ns	5
33	TdSC(INT)	Status Compare to INT Delay			ns	5,7
34	TdER(INT)	Error to INT Delay			ns	5,7
35	TdEM(INT)	Empty to INT Delay			ns	5,7
36	TdFL(INT)	Full to INT Delay			ns	5,7
37	TdS0(INT)	State 0 to INT Delay			ns	

5. [illegible]
6. [illegible] of FIO.
7. [illegible] of FIO

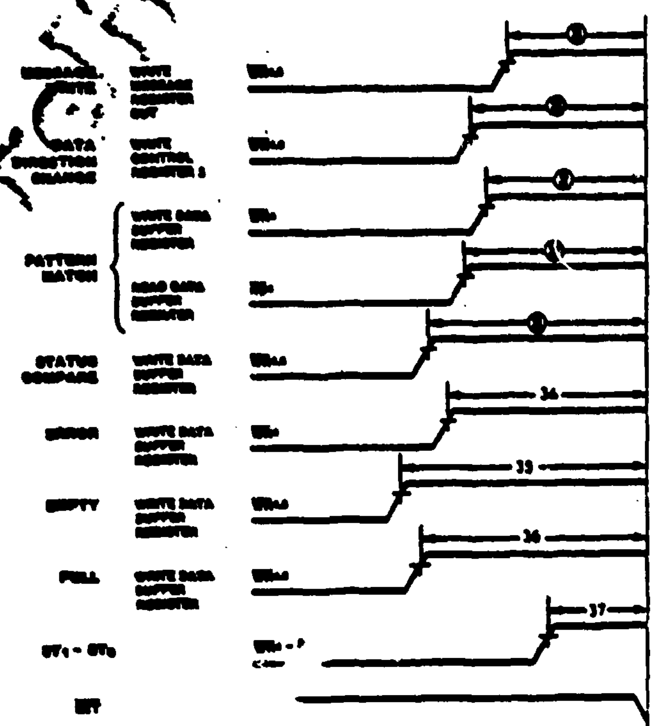


Figure 44 Non-Z-Bus Interrupt Timing

Z8038

(8)

Non-Z-Bus Request/Wait Timing

Number	Symbol	Parameter	Min	Max	Units	Notes
1	TdRD(WT)	RD↓ to WAIT Active			ns	
2	TdRD1(WT)	RD↑ to WAIT Inactive			ns	
3	TdACK(WT)	ACKIN↓ to WAIT Inactive			ns	1
4	TdRD(REQ)	RD↓ to REQ Inactive			ns	
5	TdRD1(REQ)	RD↑ to REQ Active			ns	
6	TdACK(REQ)	ACKIN↓ to REQ Active			ns	
7	TdDAC(RD)	DACK↓ to DC↓ or WR↓			ns	
8	TSU(WR)	Data Setup Time to WR			ns	
9	Th(WR)	Data Hold Time to WR			ns	
10	TdDMA	RD↓ to Valid Data			ns	2
11	TdDMA(DRH)	RD↑ to Data not Valid	0		ns	2
12	TdDMA(DRZ)	RD↑ to Data Bus Float		70	ns	2

1. The delay is from DAV↑ for 3-Wire Input Handshake. The delay is from DAC↑ for 3-Wire Input Handshake.
2. Only when DACK Inactive.

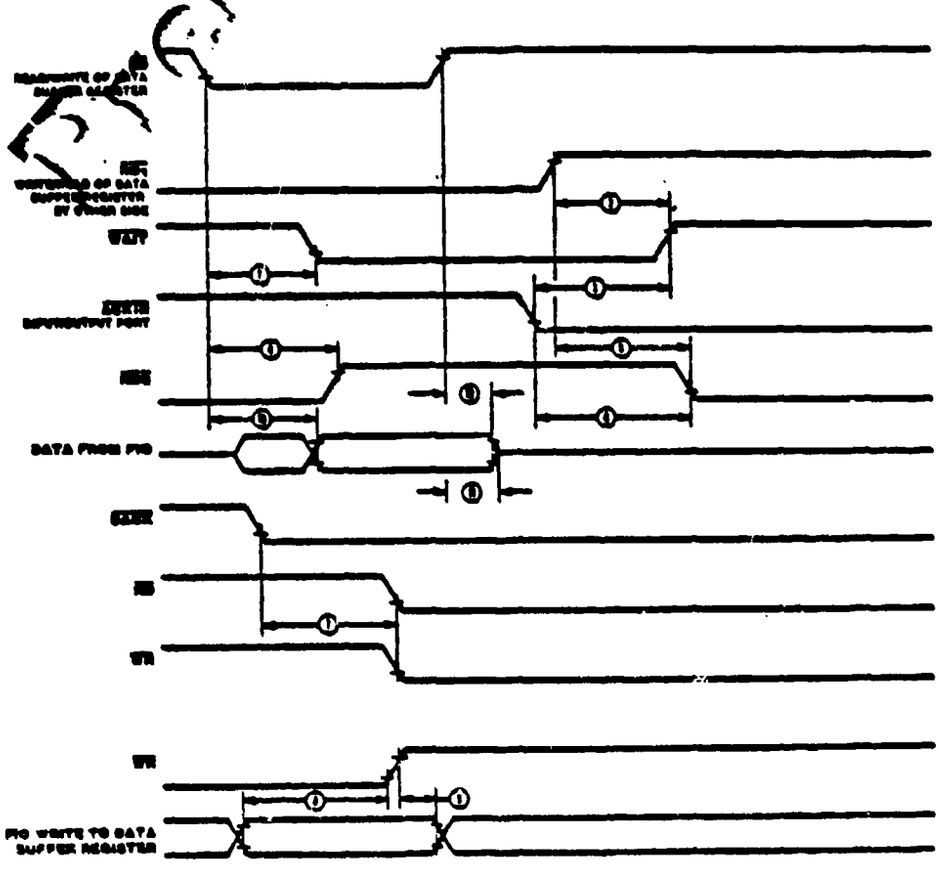


Figure 48. Non-Z-Bus REQ/WT Timing

SWITCHING CHARACTERISTICS over operating range (Notes 2, 3, 4) AM 9513 (1)

Parameter	Description		Figure	Am9513 Min	Am9513 Max	Am9513A Min	Am9513A Max	Units
TAVRL	C/D Valid to Read Low		23	25				ns
TAVWH	C/D Valid to Write High		23	170				ns
TCHCH	X2 High to X2 High (X2 Period)		24	148				ns
TCHCL	X2 High to X2 Low (X2 High Pulse Width)		24	70				ns
TCLCH	X2 Low to X2 High (X2 Low Pulse Width)		24	70				ns
TDVWH	Data In Valid to Write High		23	80				ns
TEHEH	Count Source High to Count Source High (Source Cycle Time) (Note 10)		24	148				ns
TEHEL TELEH	Count Source Pulse Duration (Note 10)		24	70				ns
TEHFV	Count Source High to FOUT Valid (Note 10)		24		500			ns
TEHGV	Count Source High to Gate Valid (Level Gating Hold Time) (Notes 10, 12, 13)		24	40				ns
TEHRL	Count Source High to Read Low (Set-up Time) (Notes 5, 10)		23	190				ns
TEHWH	Count Source High to Write High (Set-up Time) (Notes 6, 10)		23	100				ns
TEHYV	Count Source High to Out Valid (Note 10)	TC Output	24		300			ns
		Immediate or Delayed Toggle Output	24		300			
		Comparator Output	24		380			
TFN	FN High to FN+1 Valid (Note 14)		24		75			ns
TGVEH	Gate Valid to Count Source High (Level Gating Set-up Time) (Notes 10, 12, 13)		24	70				ns
TGVGV	Gate Valid to Gate Valid (Gate Pulse Duration) (Notes 11, 13)		24	145				ns
TGVWH	Gate Valid to Write High (Notes 6, 13)		23	0				ns
TRHAX	Read High to C/D Don't Care		23	0				ns
TRHEH	Read High to Count Source High (Notes 7, 10)		23	0				ns
TRHQX	Read High to Data Out Invalid		23	20				ns
TRHQZ	Read High to Data Out at High Impedance (Data Bus Release Time)		23		85			ns
TRHRL	Read High to Read Low (Read Recovery Time)		23		1000			ns
TRHSH	Read High to CS High (Note 16)		23	0				ns
TRHWL	Read High to Write Low (Read Recovery Time)		23		1000			ns
TRLQV	Read Low to Data Out Valid		23		160			ns
TRLQX	Read Low to Data Bus Driven (Data Bus Drive Time)		23	20				ns
TRLRH	Read Low to Read High (Read Pulse Duration) (Note 15)		23	160				ns
TSLRL	CS Low to Read Low (Note 16)		23	20				ns
TSLWH	CS Low to Write High (Note 16)		23	170				ns
TWHAX	Write High to C/D Don't Care		23	0				ns
TWHDX	Write High to Data In Don't Care		23	0				ns
TWHEH	Write High to Count Source High (Notes 6, 10, 17)		23	400				ns
TWHGV	Write High to Gate Valid (Notes 6, 13, 17)		23	400				ns
TWHRL	Write High to Read Low (Write Recovery Time)		23		1000			ns
TWHSH	Write High to CS High (Note 16)		23	0				ns
TWHWL	Write High to Write Low (Write Recovery Time)		23		1000			ns
TWHYV	Write High to Out Valid (Note 9, 17)		23		650			ns
TWLWH	Write Low to Write High (Write Pulse Duration) (Note 15)		23	150				ns

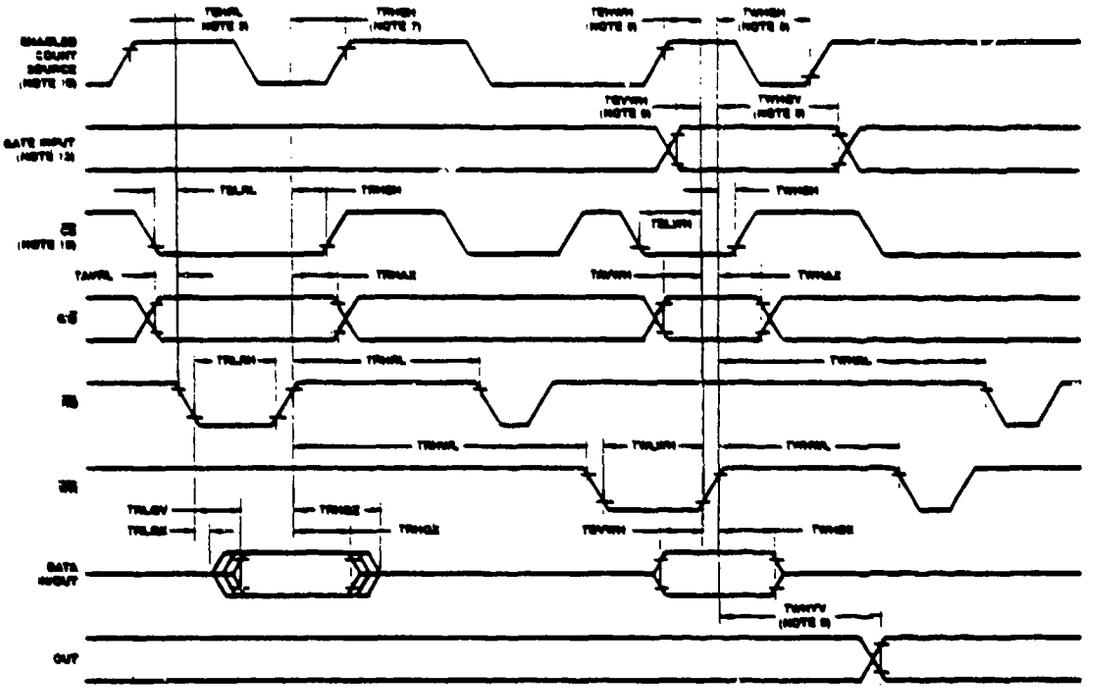


Figure 23. Bus Transfer Switching Waveforms.

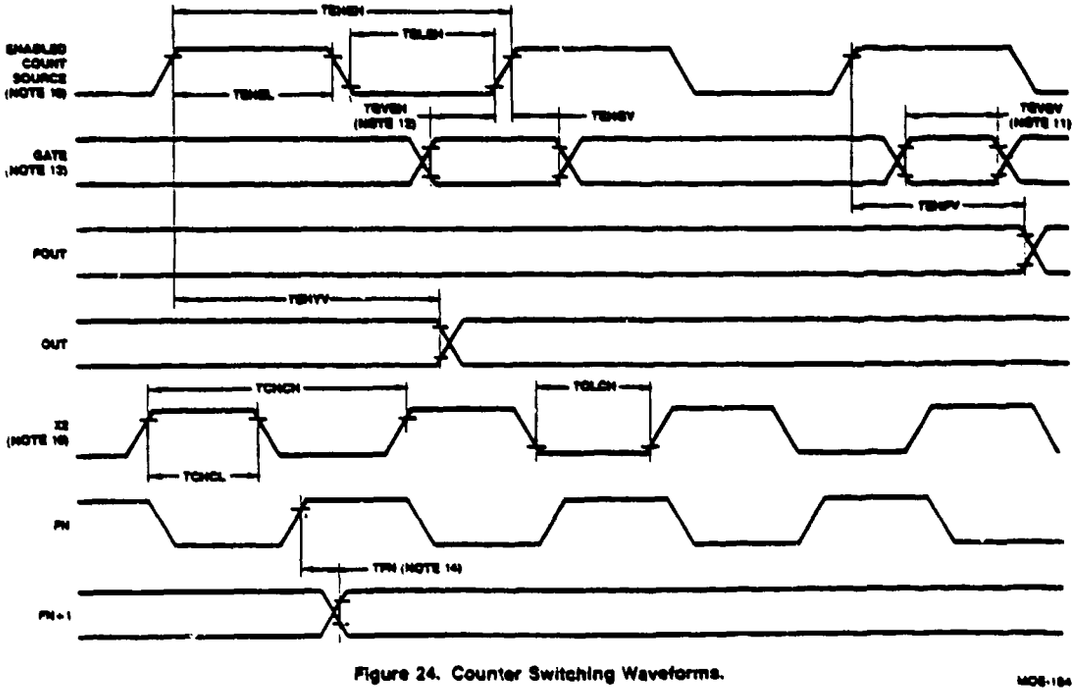


Figure 24. Counter Switching Waveforms.

SWITCHING CHARACTERISTICS Over Operating Range (Notes 2, 3, 4, 5)

Am9519

(1)

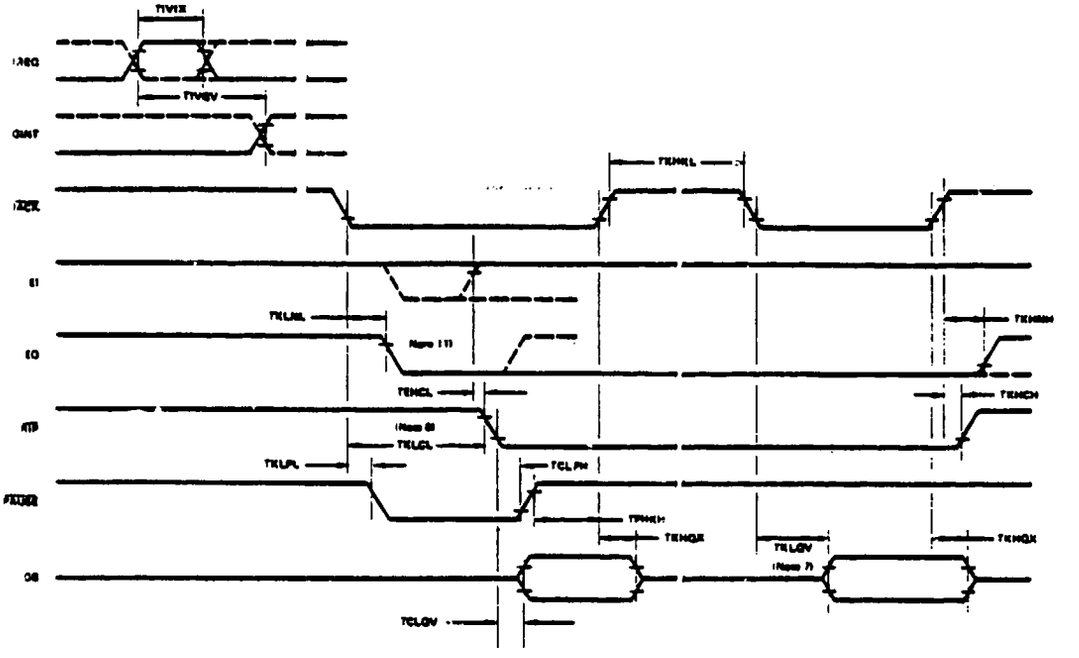
Parameters	Description	Am9519 Min.	Am9519 Max.	Am9519-1 Min.	Am9519-1 Max.	Units
TAVRL	C/$\overline{D}$ Valid and $\overline{CS}$ LOW to Read LOW	0		0		ns
TAVWL	C/$\overline{D}$ Valid and $\overline{CS}$ LOW to Write LOW	0		0		ns
TCLPH	$\overline{RIP}$ LOW to $\overline{PAUSE}$ HIGH (Note 6)	75	375	75	375	ns
TCLQV	$\overline{RIP}$ LOW to Data Out Valid (Note 7)		50		40	ns
TDVWH	Data In Valid to Write HIGH	250		200		ns
TEHCL	Enable In HIGH to $\overline{RIP}$ LOW (Notes 8, 9)	30	300	30	300	ns
TIVGV	Interrupt Request Valid to Group Interrupt Valid		800		650	ns
TIVIX	Interrupt Request Valid to Interrupt Request Don't Care (IREQ Pulse Duration)	250		250		ns
TKHCH	$\overline{IACK}$ HIGH to $\overline{RIP}$ HIGH (Note 8)		450		350	ns
TKHKL	$\overline{IACK}$ HIGH to $\overline{IACK}$ LOW ($\overline{IACK}$ Recovery)	500		500		ns
TKHNH	$\overline{IACK}$ HIGH to EO HIGH (Notes 10, 11)		975		750	ns
TKHQX	$\overline{IACK}$ HIGH to Data Out Invalid	20	200	20	100	ns
TKLCL	$\overline{IACK}$ LOW to $\overline{RIP}$ LOW (Note 8)	75	600	75	450	ns
TKLNL	$\overline{IACK}$ LOW to EO LOW (Notes 10, 11)		125		100	ns
TKLPL	$\overline{IACK}$ LOW to $\overline{PAUSE}$ LOW	25	175	25	125	ns
TKLQV	$\overline{IACK}$ LOW to Data Out Valid (Note 7)	25	300	25	200	ns
TPHKH	$\overline{PAUSE}$ HIGH to $\overline{IACK}$ HIGH	0		0		ns
TRHAX	Read HIGH to C/$\overline{D}$ and $\overline{CS}$ Don't Care	0		0		ns
TRHQX	Read HIGH to Data Out Invalid	20	200	20	100	ns
TRLQV	Read LOW to Data Out Valid		300		200	ns
TRLQX	Read LOW to Data Out Unknown	50		50		ns
TRLRH	Read LOW to Read HIGH ($\overline{RD}$ Pulse Duration)	300		250		ns
TWHAX	Write HIGH to C/$\overline{D}$ and $\overline{CS}$ Don't Care	0		0		ns
TWHDX	Write HIGH to Data In Don't Care	0		0		ns
TWHRW	Write HIGH to Read or Write LOW (Write Recovery)	600		400		ns
TWLWH	Write LOW to Write HIGH ($\overline{WR}$ Pulse Duration)	300		250		ns

NOTES:

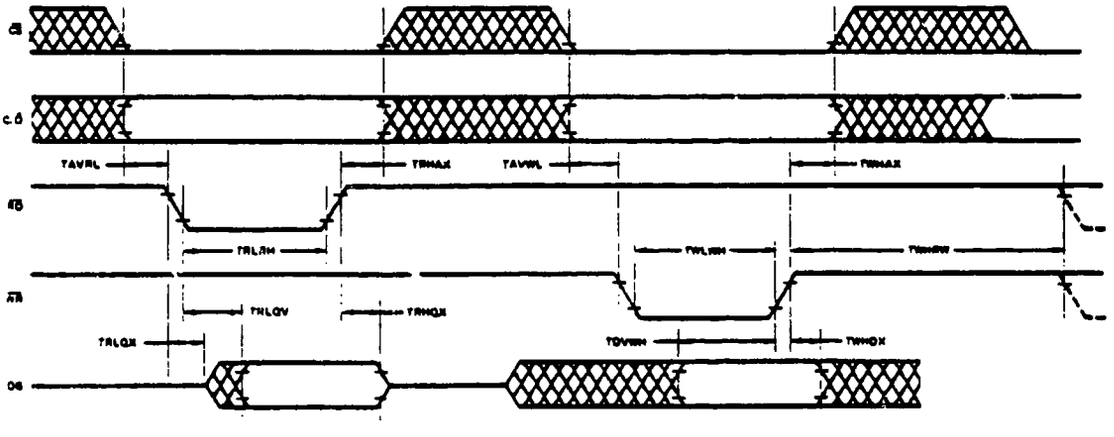
1. Typical values are for T_A = 25°C, nominal supply voltage and nominal processing parameters.
2. Test conditions assume transition times of 20ns or less, timing reference levels of 0.8V and 2.0V and output loading of one TTL gate plus 100pF, unless otherwise noted.
3. Transition abbreviations used for the switching parameter symbols include: H = High, L = Low, V = Valid, X = unknown or don't care, Z = high impedance.
4. Signal abbreviations used for the switching parameter symbols include: R = Read, W = Write, Q = Data Out, D = Data In, A = Address ($\overline{CS}$ and C/$\overline{D}$), K = Interrupt Acknowledge, N = Enable Out, E = Enable In, P = Pause, C = $\overline{RIP}$.
5. Switching parameters are listed in alphabetical order.
6. During the first $\overline{IACK}$ pulse, $\overline{PAUSE}$ will be low long enough to allow for priority resolution and will not go high until after $\overline{RIP}$ goes low (TCLPH).
7. TKLQV applies only to second, third and fourth $\overline{IACK}$ pulses while $\overline{RIP}$ is low. During the first $\overline{IACK}$ pulse, Data Out will be valid following the falling edge of $\overline{RIP}$ (TCLQV).
8. $\overline{RIP}$ is pulled low to indicate that an interrupt request has been selected. $\overline{RIP}$ cannot be pulled low until EI is high following an internal delay. TKLCL will govern the falling edge of $\overline{RIP}$ when EI is always high or is high early in the acknowledge cycle. TEHCL will govern when EI goes high later in the cycle. The rising edge of EI will be determined by the length of the preceding priority resolution chain. $\overline{RIP}$ remains low until after the rising edge of the $\overline{IACK}$ pulse that transfers the last response byte for the selected IREQ.
9. Test conditions for the EI line assume timing reference levels of 0.8V and 2.0V with transition times of 10ns or less.
10. Test conditions for the EO line assume output loading of two LS TTL gates plus 30pF and timing reference levels of 0.8V and 2.0V. Since EO normally only drives EI of another Am9519, higher speed operation can be specified with this more realistic test condition.
11. The arrival of $\overline{IACK}$ will cause EO to go low, disabling additional circuits that may be connected to EO. If no valid interrupt is pending, EO will return high when EI is high. If a pending request is selected, EO will stay low until after the last $\overline{IACK}$ pulse for that interrupt is complete and $\overline{RIP}$ goes high.
12. VOH specifications do not apply to $\overline{RIP}$ or to $\overline{GINT}$ when active-low. These outputs are open-drain and VOH levels will be determined by external circuitry.

SWITCHING WAVEFORMS

Am9519 (2)



Interrupt Operations



Data Bus Transfers

6.0 LIST OF SELECTED DATA SPECIFICATIONS

6.1 SELECTED DATA SHEETS

(1)	NMC4164	Dynamic RAM (64K x 1)
(2)	2118	Dynamic RAM (16K x 1)
(3)	NMC5295	Dynamic RAM (16K x 1)
(4)	HM-7616	Bipolar PROM (2K x 8)
(5)	2716	MOS EPROM (2K x 8)
(6)	2732	MOS EPROM (4K x 8)
(7)	SN74S225	FIFO Memory
(8)	Am25LS2518	Quad D-Type Register
(9)	Am25LS2521	Eight-bit Comparator

6.2 SELECTED PRODUCT SPECIFICATIONS

(1)	Am9513	System Timing Controller
(2)	Am9519	Universal Interrupt Controller
(3)	AmZ8127	Z8000 Clock Generator
(4)	Z80[illegible]	Serial Communication Controller
(5)	Z8038	FIFO Input-Output Interface Unit

6.3 DATA SHEETS FOR MAJOR COMPONENTS DESIGNATED FOR USE WITHIN THE PERIPHERAL ACTIVITY MODULE.

(1) IDM2910A Microprogram Controller

(2) The Am29116

APPENDIX A
DATA TABLES FOR IPPA SOFTWARE

1.0 HARDWARE STATIC MEASUREMENT BY C/T

1. preload value
2. data processing code

2.0 SOFTWARE STATIC MEASUREMENT BY SOFTWARE FUNCTION

1. frequency of collections
2. number of collections
3. processing format

3.0 EVENT

1. event name
2. definition of event
3. processing of event

4.0 STATUS

1. status name
2. definition of status (starting and ending events)
3. processing of status

5.0 PREFILLED HARDWARE STATIC MEASUREMENT AND HARDWARE CONFIGURATION TABLE

1. arm command
2. disarm command
3. preload value

4. data processing code
5. address of data
6. priority level
7. chip address
8. C/T address on chip

6.0 PAM STRUCTURE

1. CSR address
2. CSR bit patterns
3. word count register address
4. cylinder address register address
5. number of cylinders max
6. read/write to peripheral controller
7. number of NPR°s by controller and unit
8. number and role of interrupts
9. service time for transfer
10. % controller busy (MBC also)
11. disk head position/motion

7.0 PREFILLED SOFTWARE STATIC MEASUREMENT AND TIME SAMPLED SOFTWARE MANAGER CONTROL TABLE

1. measurement switch (on/off)
2. frequency
3. number of samples
4. processing format

8.0 INTERCEPTIVE MONITORING MANAGER CONTROL TABLE

1. measurement switch (on/off)
2. patch address
3. save patched executive

9.0 ARAM AND DRAM INITIALIZATION

1. starting address
2. ending address
3. bit to set

10.0 CRAM INITIALIZATION

1. start bit pattern
2. end bit pattern
3. bit to set

11.0 EVENT TABLE

1. address of event instructions

12.0 EVENT PROCESSING TABLE

1. process a
   process b
2. process a
   process c

13.0 EMT CONVERSION TABLE

1. Event number of EMT1
2. Event number of EMT2

14.0 TASK NAME CONVERSION

1. task name
2. event number
3. mask register bit setting

15.0 DATA DESCRIPTION FILE

1. measurement name
2. collected data address
3. type of measurement

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.